Discovering Clusters in Databases Using an Evolutionary Approach

by

Chung Lap Hang, Lewis

Master of Philosophy

in

Department of Computing

The Hong Kong Polytechnic University

2000

Abstract of thesis entitled 'Discovering Clusters in Databases Using an Evolutionary Approach' submitted by Chung Lap Hang, Lewis for the degree of Master of Philosophy at The Hong Kong Polytechnic University in November 1999.

Abstract

Data mining is concerned with the nontrivial extraction of implicit, previously unknown, and potentially useful information from data. The grand challenge of data mining is to collectively handle the problems imposed by the nature of real-world databases which tend to be dynamic, incomplete, redundant, noisy, and very large. Among many different problems that data mining is concerned with, the problem of discovering clusters in databases has recently received more attention. Clustering problem is concerned with the discovering of meaningful groupings of data records in a database based on their attribute values. The ability to do so can have many applications in many different areas in business and finance, computing and engineering, natural and social science, etc.

Many of the existing clustering techniques were developed to handle a special type of data mining problem called spatial data mining. The databases that are involved contain continuous-valued records and the techniques that are used are, by and large, based on distance measures that can be defined in the Euclidean space. In other words, these techniques are not very useful when employed to handle mixed continuous- and discrete-valued data records. For clustering techniques that can be used to deal with mixed data, many of them use different distance measures for continuous and discrete-valued data separately. Moreover, they are not good at handling data records that are noisy and that contain missing values. They are also not able to discover clusters whose boundaries overlap. Furthermore, many of them do not make explicit the characteristics of each cluster discovered or the differences between them and this makes the result difficult to interpret and use.

To overcome these problems, we propose a new clustering algorithm in this thesis. When compared with existing algorithms, it has several advantageous features. It is able to (i) handle mixed continuous- and discrete-valued data; (ii) discover overlapping clusters; (iii) perform data transformation; (iv) handle noisy and missing values; and (v) explicitly represent the characteristics of each discovered clusters.

The proposed clustering algorithm is based on the use of a simple genetic algorithm (GA). By representing a cluster label as a gene, a particular grouping of records is encoded in a chromosome. Once different groupings are generated, the most interesting chromosomes are then evolved using the operators of selection, crossover, and mutation. To determine how interesting the chromosomes in the whole population is, all of them are evaluated by a fitness function. The fitness function is defined in terms of a probabilistic similarity measure and can be interpreted as an objective measure of interestingness of the rules that characterize the particular grouping in a chromosome. Since the similarity measure is probabilistic, it can be defined even when the data being dealt with contains noisy, dynamic, incomplete, missing, or even erroneous values. In addition, the defined measure enables us to discover overlapping clusters. The ultimate goal of the evolutionary process is, therefore, to identify the fittest chromosome by maximizing the fitness function. During the process, it should be noted that a set of rules is discovered to characterize the specific grouping encoded in a chromosome. This non-black-box approach makes it possible for the patterns underlying the databases to be made explicit.

To evaluate the performance of the proposed clustering approach, we used many sets of real and simulated data in our experimentation. In addition, for comparison with existing methods, we have also introduced an evaluation criterion. The results of the experiments show that the proposed approach is able to handle real problem more effectively.

Acknowledgements

Although on the cover of any thesis only the name of the author appears, a thesis is always the result of the blood, sweat, and tears of many people. I would like to express my sincere thanks to my supervisor, Dr. Keith C.C. Chan, for providing constructive comments and advice, innovative ideas, and continuous support in my graduate study at The Hong Kong Polytechnic University. Without his supervision, my research study is hardly to complete. It's really a great time to work with Keith as I've learned many valuable things from him that couldn't be learned from books. These treasures will surely be useful in my life.

I would like to thank all my colleagues who provided encouragement and a nice working environment. In particular, I would like to thank Lawrance Chung and Vincent Li for keeping my spirits up. Most importantly, I would like to thank all my family members for their patience, understanding and support.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

With the proliferation of powerful and affordable computing and data gathering devices, we have seen a dramatic increase in the amount of electronic data being collected and stored. It has been estimated that the amount of electronic data in the world doubles every 20 months, and the size and number of databases that store the data are increasing even faster [Marshall 1999]. For example, one of the world's largest retailers, Wal-Mart, as of 1995, had 65 weeks of point-of-sale transaction data on-line, taking up more than 3.5 terabytes of storage [Hedberg 1995]. Also, Mobile Oil Corporation has been developing a data warehouse capable of storing terabytes of data related to oil exploration; and many health-care companies in the U.S. have stored their transactions on computers, yielding multi-gigabyte databases [Fayyad 1996; Fayyad, Piatetsky-Shapiro, and Smyth 1996a].

Given that we have vast amounts of data stored and available to us, and that new mountains of data are being gathered every day, many businesses and organizations are now facing with the problem of how best to make use of all these data. It has, therefore, resulted in a need for new techniques and tools to intelligently and automatically analyze them to obtain the knowledge hidden in them and thus applications such as information management, query processing, decision making, process control, etc. can be better performed [Chen, Han, and Yu 1996]. It is perhaps for the potentially widespread applications that data mining has been considered one of the most important topics of database research [Silberschatz, Stonebraker, and Ullman 1991, 1996].

Data mining is concerned with the nontrivial extraction of implicit, previously unknown, and potentially useful information from data [Frawley, Piatetsky-Shapiro, and Matheus 1991]. It is one of the steps in the process of knowledge discovery in databases (KDD) [Fayyad 1996; Fayyad, Piatetsky-Shapiro, and Smyth 1996a, 1996b, 1996c]. And for this reason, data mining has been used interchangeably with KDD by many database researchers [Agrawal *et al.* 1996; Han *et al.* 1996; Imielinski and Virmani 1995; Silberschatz, Stonebraker, and Ullman 1996].

The grand challenge of data mining is to collectively handle the problems imposed by the nature of real-world databases which tend to be dynamic, incomplete, redundant, noisy, and very large [Matheus, Chan, and Piatetsky-Shapiro1993]. Many interesting studies on data mining to draw upon methods, algorithms, and techniques from fields as diverse as machine learning, pattern recognition, database systems, statistics, artificial intelligence, knowledge acquisition, and data visualization have been carried out [Fayyad *et al.* 1996; Piatetsky-Shapiro and Frawley 1991]. Roughly, data mining techniques can be classified according to the kind of knowledge they mine for such as association, sequencing, classification and clustering.

Association aims at discovering interesting relationships or associations among different attributes. (e.g. [Agrawal, Srikant 1994; Agrawal, Imielinski, and Swami 1993; and Agrawal *et al.* 1996]). Similarly, sequential analysis also attempts to discover association in time series data. The major difference between association and sequencing is that the former discovers patterns inside a record whereas the later discovers patterns between records. Because of this characteristic, they are usually applied in transaction databases.

Given a set of pre-classified data, the classification problem (e.g. [Agrawal *et al.* 1992; Lu, Setiono, and Liu 1995, 1996]) is concerned with finding a classifier usually in the form of a set of rules, that is able to allow data records to be classified into different predefined classes. The classification problem has been studied extensively by researchers in the machine learning community (see, e.g. [Michie, Spiegelhalter, and Taylor 1994]) and various techniques have been developed to solve it. Among them, the decision-tree based techniques are the most popular (e.g. [Agrawal *et al.* 1992; Lu, Setiono, and Liu 1995, 1996]).

The mining of classification rules requires training samples and the mining problem is considered supervised. If training samples are not provided, the discovery of grouping of records becomes unsupervised and such problem is called a clustering problem. The clustering task (e.g. [Ng, Han 1994; Ester, Kriegel, Sander, Xu 1996; Zhang, Ramakrishnan, and Livny 1996; Michaud 1997; Huang 1997a, 1997b, 1998; Li and Biswas 1997; Biswas, Weinberg and Fisher 1998; Ramkumar and Swami 1998; and Han *et al.* 1998]) aims at discovering meaningful grouping of records which is not associated with any predefined classes. To do so, clustering techniques should perform their task from a set of records based on their attribute values by maximizing the intra-class and, at the same time, minimizing the interclass similarities [Anderberg 1973; Jain and Dubes 1988; Everitt 1993]. Features in common in the data records are identified and grouped together. Clustering problem is vital in data mining applications because clustering algorithm is able to discover meaningful grouping without the need of supervision.

Among the different problems data mining is concerned with, the problem of clustering [Ng, Han 1994; Ester *et al.* 1996; Zhang, Ramakrishnan, and Livny 1996; Michaud 1997; Huang 1997a, 1997b, 1998; Li and Biswas 1997; Biswas, Weinberg and Fisher 1998; Ramkumar and Swami 1998; and Han *et al.* 1998] has recently received more attention. This is mainly because a majority of databases cannot be predefined meaningfully due to the lack of domain knowledge. In addition, a good clustering algorithm can be applied to many applications in many different areas. For example, clustering of medical data can allow patients that require the same treatment methods to be grouped together for more efficient services; clustering of supermarket data can allow sale items that are often purchased together to be grouped for shelf-space organization; and clustering of customer data of a bank permits the bank to segment its customers for future marketing plan [Apte 1997]. In this thesis, we are, therefore, concerning with this task.

1.1 The Problems

The problem of clustering can be described as follows. Given a database, D containing N records, R_1, ..., R_q, ..., R_N, with q = 1, 2, ..., N. Suppose that each record in the database is described by n distinct attributes, A_1, ..., A_j, ..., A_n, with respective domains of $dom(A_1)$, ..., $dom(A_j)$, ..., $dom(A_n)$, A_j can take on continuous or discrete data values. Suppose also that the data are noisy in the sense that a certain percentage of values are incomplete, inconsistent or incorrect so that the patterns inherent in the data are not completely deterministic. The problem we are concerned with is to discover the meaningful grouping from the given database, D into clusters without any a priori knowledge about the data and without the need to assume that $\bigcup_{k=1}^{K} C_k = \{R_1, R_2, \ldots, R_n\}$ and $C_k \cap C_l = \emptyset$, $k \neq l$ where $C_k = \{R_q \mid q \in \{1, 2, \ldots, n\}\}$ and $k = 1, 2, \ldots, K$ and K are the total number of discovered clusters. We assume that a record can belong to one or more than one cluster.

Given the definition, several sub-problems have to be addressed: (1) handling of transaction and relational (continuous- and discrete-valued) data together; (2) discovering of overlapping clusters; (3) data transformation; (4) handling of noisy and missing values; and (5) interpretation of clustering result.

Recently, data mining researchers have been focusing on clustering involving continuous-valued data [Ng, Han 1994; Ester *et al.* 1996; and Zhang, Ramakrishnan, and Livny 1996] and some effective approaches have been developed. In [Brachman *et al.* 1996; Fayyad, Piatetsky-Shapiro, and Smyth 1996c; and Fayyad 1997; and Bhandari, *et. al.* 1997], some applications on clustering continuous-valued data have been described. Given two records R_i and R_j, the similarity between R_i and R_j is ultimately determined by the distance between them for many of the existing algorithms [Ng, Han 1994; Ester *et al.* 1996; and Zhang, Ramakrishnan, and Livny 1996]. The larger the distance between R_i and R_j, the lesser their similarity and the more their dissimilarity. The goal of many of these clustering algorithms is to maximize or minimize a function f, which is based on a distance measure. With the use of such function, a weakness of these techniques is that they cannot handle both continuous- and discrete-valued data at the same time. It is because different distance measures are required to handle different types of data. For example, Euclidean distance is often used to handle continuous-valued data whereas Hamming distance is often used to handle discrete-valued data. Applying arithmetic operations on two different distance measures may not be meaningful.

It is not unusual that there are many database systems contain not only relational data (such as customer information, inventory records, and credit histories, etc.) but also transaction data (such as records of purchase, electronic fund transfer or phone calls, etc.). These data very often consist of both continuous- and discrete-valued data. However, the clustering task in data mining (e.g. [MacQueen 1967; Ward 1963; Ng; Han 1994; Ester, Kriegel, Sander, Xu 1996; Zhang, Ramakrishnan, and Livny 1996]) is mainly developed for revealing those clusters in continuous-valued data which are always referred to as spatial data. Because of this, none of these algorithms is explicitly developed for handling relational data and furthermore, it is not clear how these algorithms, which only deal with spatial data, can be applied to discover clusters relating the union of both transaction and relational data. For example, cluster characterized by "Female customers aged between 30 to 40 with large purchase amount in the 2nd quarter of a year" cannot be discovered by existing algorithms. These descriptions relate to grouping patterns of the sales transaction system. To discover patterns of such kind, both of transaction and relational data have to be taken into consideration. Since it is common for both transaction and relational data to be collected in many database systems, it is important that this problem be dealt with effectively. Our goal is to discover the grouping of such records and to provide explicitly a set of rules describing the interesting patterns inside clusters.

Besides the problem in handling a mix of both continuous- and discrete-valued data, existing clustering algorithms [Ng, Han 1994; Ester *et al.* 1996; Zhang, Ramakrishnan, and Livny 1996; Michaud 1997; Huang 1997a, 1997b, 1998; Li and Biswas 1997; Biswas, Weinberg and Fisher 1998; Ramkumar and Swami 1998; and Han *et al.* 1998], whether or not they are able to handle both continuous- and discrete-valued data and whether or not they employ a distance metric for similarity measure, do not allow a record to belong to more than one cluster. In other words, they are unable to discover clusters whose boundaries overlap. However, it is possible that a record may belong to one or more than one cluster. For example, given a medical database, assume we have two clusters mainly characterized by flu and headache respectively. If we now have a patient who got both flu and headache at the same time, it is quite difficult to justify this patient to be assigned only to either one of the clusters. Clustering algorithm for data mining tasks would be more effective if it is able to discover overlapping clusters.

Data transformation has been addressed and considered as one of the essential steps in data mining [Fayyad, Piatetsky-Shapiro, Smyth, and Uthurusamy 1996; and Fayyad, Piatetsky-Shapiro, and Smyth 1996a; and Fayyad, Piatetsky-Shapiro, and Smyth 1996b]. It has been studied as part of data warehousing. Unfortunately, existing data mining techniques do not provide any explicit methodology for data transformation.

Data transformation is to transform data so that more interesting patterns can be discovered by data mining algorithms. Undoubtedly, without using data transformation, some useful and important patterns representing the data are unable to be found. It is also not possible to discover patterns involving attributes not originally contained in the database being mined. For example, cluster characterized by the phone calls that are over one hour in duration and are made during Christmas by users whose monthly payments average less than fifty dollars cannot be discovered as the database does not contain explicitly the attribute values of "Christmas" and "average monthly payment". These attributes are functions of the "date of call" and "monthly payment" and are not stored in the original transaction and relational data. In order to discover such kind of interesting patterns, one must calculate the charge of each phone call based on the start time, the end time, and the charge per minute for that call period. Without data transformation, useful and important patterns may not be utilized in the data mining tasks. As a result, meaningful grouping with interesting rules may not be discovered even with the most effective mining technique.

Inside huge amount of data for knowledge discovery process, noisy data and missing attribute values always exist [Frawley, Piatetsky-Shapiro, and Matheus 1991; Fayyad *et al.* 1996; Fayyad, Haussler, and Stolorz 1996a, 1996b; and Fayyad, Piatetsky-Shapiro, and Smyth 1996b]. For example, U.S. census data reportedly has error rates of up to 20%. In addition, useful patterns inside data may not be deterministic. For example, it is hardly to find two attribute values with perfect relationship (i.e. the existence of an attribute value is 100% with another attribute value). Instead of such relationship, it is common to find that only a certain percentage of records characterized by an attribute value and also by another attribute value. Clustering task should therefore be able to handle data probabilistically.

With noisy and missing values in data, they can obscure the underlying pattern and cause confusion, especially if a key input variable is noisy. Even data cleaning has been discussed to handle noisy data [Fayyad, Piatetsky-Shapiro, and Smyth

1996a, 1996b; Fayyad 1998a], however, in some cases, performing data cleaning may be theoretically incorrect for a certain problem domain. For example, given a clinical database, some features may only exist in male patients but not in female patients. In this case, the missing values of such features in female patients actually contain some information instead of noise. Substituting any values to the missing values is obviously incorrect. Among typical data cleaning and preprocessing techniques such as inserting normalized data to the missing value, modifying prior probabilities, averaging data values, etc. [Kennedy *et al.* 1997], they may incur a gross distortion of the original data [Matthews and Hearne 1991]. In addition to this problem, users may not have any idea and knowledge to apply which kind of data cleaning techniques. Therefore, a data mining technique that can handle noisy and missing values will be an advantage for the data mining process.

No matter what knowledge has been mined, it is important to present the discovered knowledge in a novel and easy understandable way to human. Data interpretation is a step for such task [Fayyad, Piatetsky-Shapiro, and Smyth 1996a, 1996b, 1996c]. However, for those clustering techniques that are based on a distance metric (e.g. [Forgy 1965; MacQueen 1967; Michaud 1997; Huang 1997a; Huang 1997b; Huang 1998; Fayyad, Reina, and Bradley 1998, Ng, Han 1994; Ester *et al.* 1996; and Zhang, Ramakrishnan, and Livny 1996]), they are only able to produce the result about which cluster that objects or records belong to. Since these clustering algorithms can only determine the degree of similarity between two objects, they do not provide us the meaning about how these objects or records are grouped together. In other words, what knowledge has been mined from these clustering techniques is not clearly known. In addition, some of these clustering techniques are effective only if the data given are all continuous-valued data and the number of dimensions is small such as spatial data (e.g. [Ng, Han 1994; Ester *et al.* 1996; and Zhang, Ramakrishnan, and Livny 1996]). In this case, the clustering result can then be easily visualized in a 2-D plot graph. With a set of data from transaction and relational databases which contain both continuous- and discrete-valued data, such techniques may, however, not be very suitable in the sense that clustering result is much more difficult or even unable to be visualized in a graph.

For those clustering techniques that are not based on a distance metric (e.g. [Michalski 1980; Fisher, Pazzani and Langley 1991; Fisher 1987; Fisher *et al.* 1993; Anderson and Matessa 1991; and Gennari, Langley and Fisher 1990; and Cheesman and Stutz 1995]), many of them [Michalski 1980; and Cheesman and Stutz 1995] do not produce any interpretation of the clustering result. Although there are approaches [Fisher, Pazzani and Langley 1991; Fisher 1987] which are able to generate a hierarchy, it is still, to a certain extent, quite difficult to understand patterns inside the hierarchy. To be able to understand how one cluster is different from another, a separate inductive learning algorithm such as c4.5 [Quinlan 1993] which is normally designed quite differently from the algorithm that generates the clusters may have to be used. The rules induced may therefore not quite describe a cluster perfectly.

To represent the discovered patterns, if-then rule associated with an interesting measure is one of the representative methods and it has been widely applied in mining association rules [Agrawal, Srikant 1994; Agrawal, Imielinski, and Swami 1993; and Agrawal *et al.* 1996] with great success. Among many interesting measures judging the degree of confidence for each description of mined result, the *support* and *confidence* measure [Agrawal, Srikant 1994; Agrawal, Imielinski, and Swami 1993; and Agrawal *et al.* 1996] is typically used. The rule is interesting if and only if its *support* and *confidence* values are greater than some user-supplied threshold. A weakness of such approach is that many users do not have any idea what the threshold should be. If it is set too high, a user may miss some useful rules but if it is set too low, the user may be overwhelmed by many irrelevant ones. It would surely be better to have an objective interestingness measure for each description of result when if-then rules are employed so that the knowledge presented would be more reliable.

To perform more effectively, data mining techniques for clustering tasks should be improved in some areas including (i) the handling of both transaction and relational data instead of solely spatial data; (ii) the discovery of overlapping clusters; (iii) the performing of data transformation; (iv) the handling of noisy and missing values in a probabilistic data environment; and (v) the interpretation of results for clustering. In this thesis, we propose a method that is equipped with the capabilities to tackle problems in these areas.

1.2 Overview of solution

To overcome the problems we introduced in the last section, our proposed clustering algorithm uses a simple genetic algorithm (GA). One of the primary advantages of the GA is that it is application domain independent. This can thus be applied to different types of applications and different problem domains. For a GA to be applied effectively, one must consider the followings: (i) encoding which encodes the solutions to a problem in a chromosome and (ii) evaluation which measures the worth of any chromosome on the problem to be solved. With these two mechanisms, a GA searches for the best solution in the solution space by using some genetic operators (crossover and mutation) to manipulate genetic materials. Our algorithm uses a GA called steady-state genetic algorithm (SSGA) to solve the clustering task. It differs from simple GA in such a way that the whole population is not completely deleted for each generation and thus the best chromosomes would not be lost. By representing a cluster label as a gene, a particular grouping of records is encoded in a chromosome. Through an evolutionary process, it is able to search for one of the most interesting grouping of records among all possible groupings stochastically. To do so, several genetic operators are used to reproduce new chromosomes in the evolutionary process. They are the uniform list crossover and uniform list mutation. These two operators have an advantage over typical one-point crossover and mutation in such a way that the location of the encoding of a feature on a chromosome is irrelevant.

With these genetic operators, the goal of the evolutionary process is to find the chromosome that is the fittest according to the fitness function. The fitness function that we defined is in terms of a probabilistic similarity measure. Since the fitness is probabilistic, it can be defined even when the data being dealt with contains noisy, dynamic, incomplete, missing, or even erroneous values. For clustering task to be performed, algorithm should be able to maximize the intra-class similarity and, at the same time, be able to minimize the inter-class similarity. The similarity in our fitness function can also be derived in terms of the *support* and *confidence* in mining association rules [Agrawal, Imielinski, and Swami 1993, Agrawal, Srikant 1994, and Agrawal *et al.* 1996]. Before the overall fitness value is determined, our algorithm is able to identify interesting patterns from a specific grouping of records in a chromosome. Unlike the use of the *support* and *confidence* in mining association rules, the extraction of such interesting patterns is based on an objective measure. Therefore, patterns can be uncovered without using any user-supplied threshold. The greater the interestingness reflected by fitness value, the fitter and the more interesting and meaningful grouping the chromosome encodes. The most interesting grouping in a chromosome is thus by maximizing the defined fitness function. Moreover, our algorithm is able to discover overlapping clusters through the fitness function. Furthermore, it is also able to handle both continuous- and discrete-valued data without the need of domain knowledge. To aid interpretation of the clustering results, our algorithm is able to generate a set of rules when it performs its task. The rules can describe why a record belongs to a particular cluster. This non-black-box approach makes it possible for the patterns underlying the databases to be made explicit.

1.3 Outline of Thesis

In the rest of this thesis, we present the details of our algorithm. In Chapter 2, we give a survey of related work in data mining in solving the clustering problem. Specifically, four approaches are to be described. They are the traditional approaches, the artificial intelligence approaches, the neural network approaches, and the data mining approaches for clustering. In addition to the survey on these approaches, the difficulties and weaknesses of how these approaches apply to our problem are also described.

In Chapter 3, we describe a data mining approach that makes use of a simple genetic algorithm called steady-state genetic algorithm (SSGA) for the clustering task. Our algorithm encodes a particular grouping in a chromosome so that each gene represents a group label. Through an evolutionary process, it is able to search for one of the most interesting grouping of records among all possible groupings stochastically. To do so, a fitness function is defined for such process. The fitness that we defined is in terms of a probabilistic similarity measure. For clustering task to be performed, algorithm should be able to maximize the intra-class similarity and, at the same time, be able to minimize the inter-class similarity. Our fitness function defined can perform clustering in this way and can also be derived from the *support* and *confidence* values [Agrawal, Imielinski, and Swami 1993; Agrawal, Srikant 1994; and Agrawal *et al.* 1996] which are a very popular component for association task in data mining. In addition to the use of the *support* and *confidence* values, our fitness function is based on an objective measure to identify interesting pattern hidden in the data. Based on such objective measure, it is able to uncover the interesting patterns without using any user-supplied thresholds. With maximizing the defined fitness, the greater the interestingness, the fitter and the more interesting and meaningful grouping of the chromosome encodes. The goal of the evolutionary process is to find the chromosome that is the fittest. It is undoubtedly that data used for data mining are always be noisy, dynamic, incomplete, missing, and erroneous. Therefore, an effective clustering technique for data mining should be able to handle data even in a noisy environment. Our approach is able to perform its task in such environment. Moreover, it is able to discover overlapping clusters. Furthermore, our clustering approach is able to handle both continuous- and discrete-valued data without the need of domain knowledge. To aid the data interpretation in data mining process, our approach is able to generate a set of rules during it performs its task. The rules can describe why a record belongs to a particular cluster.

To evaluate the performance of our method, we used several simulated and real-life databases for our experiments presented in Chapter 4. Before presenting the evaluation, a description of all evaluated clustering algorithms is given. Finally, a discussion on the experimental results and the problems and difficulties of the evaluated algorithms in applying to data mining are also described.

Although the algorithm proposed in Chapter 3 could be used to discover clusters and interesting interpretation for each cluster, it can be further improved by incorporating transformation functions. The use of transformation functions allows us to handle relational, transaction and the union of the transaction and relational data. In Chapter 5, we then introduce a formalism of this problem and propose a novel solution for it. Specifically, we define a set of transformation functions for different types of attributes in both the transaction and relational databases. Based on the transformation functions, we are able to perform data transformation which results in a set of new relations so that interesting patterns can be discovered from these transformed relations.

We conclude our study in Chapter 6. The major work carried out in our study is summarized. We also provide some possible extensions to our work and discuss some potential research problems in the future.

Chapter 2

Survey of Related Work

In this chapter, we will give a literature survey on clustering. The research on clustering has been widely studied for many years. We describe all different types of clustering techniques to solve the problem. It can be generally classified into the following categories: (1) the traditional approaches; (2) the artificial intelligence approaches; (3) the neural network approaches; and (4) the data mining approaches. Their relative advantages and disadvantages are discussed.

2.1 The traditional approaches

Cluster analysis has been widely studied in statistics in the past 30 years. One of the oldest approaches is used for numerical taxonomy. The identification of clusters is by means of grouping data objects that are similar to one another into one cluster. Similarity is always expressed in terms of a distance function, which is typically, though not necessarily, a metric [BarBara *et al.* 1997]. For example, for each pair of data objects p_1, p_2, the distance $D(p_1,p_2)$ is known. In addition to a distance function, there is a separate "quality" function that measures the "goodness" of a cluster. One example of a quality function is the centroid distance, i.e., the average over $\{D(p_1,c) \mid p_1 \in C_1\}$, where c is the centroid of all the objects in cluster C_1. Another example is the diameter, i.e., the maximum over $\{D(p_1,p_2) \mid p_1,p_2 \in C_1\}$. Even though similarity between objects and goodness of clusters can be defined, it is much harder to define "similar enough" and "good enough". The answer to this question is typically highly subjective and remains an open issue in cluster analysis [Kaufman and Rousseeuw 1990].

In traditional clustering approaches, one classification partitions the techniques on the basis of the type of control used in building the clusters. The categories of clustering techniques according to this classification are (i) hierarchical methods; (ii) partitioning or optimization methods; (iii) density or mode-seeking methods; and (iv) clumping methods. All of these methods are well-known traditional methods for clustering.

2.1.1 Hierarchical clustering methods

Hierarchical clustering is perhaps the oldest heuristic approach to perform clustering. The techniques may be subdivided into agglomerative methods which proceed by a series of successive fusions of the *n* objects into groups, and divisive methods which partition the set of *n* objects successively into finer partitions. The results of both agglomerative and divisive techniques may be presented in the form of a dendrogram, which is a two-dimensional diagram illustrating the fusions or partitions which have been made at each successive level. Fig. 1 gives an example of a dendrogram.

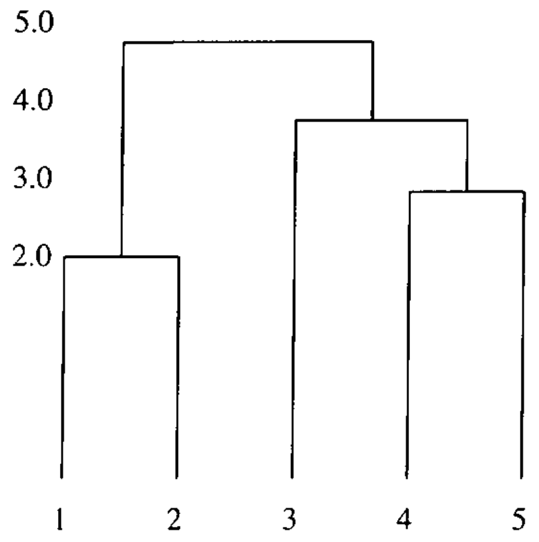


Figure 1. An example of a dendrogram [Everitt 1993]

Both types of hierarchical technique may be viewed as attempts to find the most efficient step, in some defined sense, at each stage in the progressive subdivision or synthesis of the population. One of the primary drawbacks of hierarchical technique is that the divisions or fusions once made are irrevocable. Also, hierarchical clustering requires to determine manually the stopping point during subdivision or synthesis of population [Anderberg 1973; Jain and Dubes 1988; and Everitt 1993]. It is because all agglomerative hierarchical techniques ultimately reduce the data to a single cluster containing all the objects, and the divisive techniques will finally split the entire set of data into *n* groups each containing a single entity. The following gives the details of the two hierarchical clustering techniques.

Agglomerative techniques

As stated before, agglomerative techniques form clusters by progressive fusion, that is, by recursively joining separate objects and small groups together to form larger and larger groupings [Anderberg 1973; Jain and Dubes 1988; and Everitt 1993]. Eventually a single universal group is formed and the process halts, leaving a record of the merges that took place. The history of merges is often displayed in the form of a dendrogram that shows, by the position of the horizontal location of the merge, the between-group similarities. As the groups encompass more and more objects, the between-group similarity scores decrease.

By adopting a threshold of minimum similarity, the agglomeration process can be halted before all objects are merged into a single group. Besides the threshold, the stopping criteria may be also a reasonable number of clusters, which may be suggested from knowledge from the domain of application, or when the next merge causes a relatively large increase in the objective function value. Conversely, the complete dendrogram may be "cut" apart across some similarity boundary. This yields a number of clusters, each containing those objects that were merged at a similarity score above the given threshold.

During the agglomerative clustering process, it is necessary to calculate the similarities between groups of objects. This can be achieved by the computation of a distance matrix between the objects. For example, one of the most common distance measures is Euclidean distance. There are also other distance measures like City block distance and angular separation. After obtaining the distance matrix, the between-group similarities can be measured as the reciprocal of distances. There are many standard methods to measure the similarities. Below gives a briefly description of this issue.

The nearest neighbor or single link method

Groups initially consisting of single individuals are fused according to the distance between their nearest members, the groups with the smallest distance being fused. Each fusion decreases by one the number of groups. For this method, then, the distance between groups is defined as the distance between their closest members.

The furthest neighbor or complete linkage method
The method is exactly the opposite of the single linkage method, in that distance between groups is now defined as the distance between their most remote pair of individuals.

Centroid cluster analysis
In centroid cluster analysis, groups lie in Euclidean space, and are replaced on formation by the co-ordinates of their centroid. The distance between groups is defined as the distance between the group controids. The procedure then is to fuse groups according to the distance between their centroids, the groups with the smallest distance being fused first.

Ward's method
Ward [Ward 1963] proposed that at any stage of an analysis the loss of information which results from the grouping of individuals into clusters can be measured by the total sum of squared deviations of every point from the mean of the cluster to which it belongs. At each step in the analysis, union of every possible pair of clusters is considered and the two clusters whose fusion results in the minimum increase in the error sum of squares are combined.

Although agglomerative hierarchical clustering is widely adopted in clustering analysis, it suffers a quite high time complexity drawback. Naive implementations of agglomerative hierarchical clustering have computational complexity $O(n^3)$; but $O(n^2)$ implementations have been developed [Murtagh 1985]. The approach may still be impractical for very large problems such as data mining tasks.

Divisive techniques
In contrast to agglomerative procedures, divisive hierarchical clustering starts by considering the entire set of elements as a single cluster, and iteratively splits one cluster into two until each element is in its own cluster or until some other stopping criterion is satisfied. The result is also a hierarchy of objects. The divisive technique of Edwards and Cavalli-Sforza [Edwards, and Cavalli-Sforza 1965] examines all $2^{n-1}-1$ partitions of n objects and selects the one that gives the minimum intracluster sum of the squared interobject distances. The computational cost of the method limits its use to cases involving the clustering of only a few objects. Divisive

methods tend to be less widely used than agglomerative methods, because they cannot recover from a poor choice in the early splits.

2.1.2 Partitioning or optimization methods

The partitioning or optimization methods, also known as the direct methods in [Shapiro and Stuart 1992], neither merge objects into clusters nor break large clusters into smaller ones. These techniques are given the number (usually denoted k) of clusters to form and proceed to find a partitioning of the objects into k clusters that optimize some measures of the goodness of the clusters. The measure of the goodness is referred to as the 'clustering criterion'. Having the clustering criterion, three distinct procedures can be taken so as to maximum or minimum the defined criterion: 1) initiating clusters; 2) allocating objects to initiated clusters; and 3) reallocating some or all of the objects to other cluster once the initial classificatory process has been completed.

One of the early direct clustering techniques is k-means developed by MacQueen [MacQueen 1967]. Other variations of k-means can be found in [Anderberg 1973; and Spath 1980]. The optimization method can also be separated into the monothetic and the polythetic techniques. A monothetic clustering algorithm divides the set of objects into clusters that differ in the value of one attribute. For example, such a technique might form one cluster in which attribute X_i has the value 1 and another cluster in which attribute X_i has the value 0. A polythetic clustering technique forms clusters in which the values of several attributes differ for different classes.

K-means procedure

A widely used clustering procedure searches for a nearly optimal partition with a fixed [Forgy 1965] number of clusters. First, an initial partition with the chosen number of clusters is built (it can be done in many ways). Then, keeping the same number of clusters, the partition is improved iteratively. Each element is handled sequentially and reassigned to the cluster such that the partitioning criterion is most improved by the reassignment. This is a "minor iteration"; a sequence of n minor iterations, one for each element, is a "major iteration". Usually the procedure ends when no improving reassignment is obtained in a major iteration, but stronger end tests can also be used.

Since the *k*-means procedure works on a fixed number of clusters, one needs to repeat the procedure with different numbers of clusters for a final solution. If only a small set of numbers of clusters is searched, the computation requirement is essentially linear in *n*.

2.1.3 Density or Mode-seeking methods

In traditional clustering approaches, some researchers suggested that clusters can be identified by high density region from low density region. The method based on this idea is called density or mode-seeking method that clusters are formed by searching for regions containing a relatively dense concentration of objects. If objects are depicted as points in a metric space, a natural concept of clustering suggests that there should be parts of the space in which the points are very dense, separated by parts of low density. There are many methods of cluster analysis which use this approach of seeking regions of high density or modes in the data. In general, each mode is taken to signify a different group.

Several of these methods have their origins in single linkage cluster analysis and arise in an attempt to overcome the main problem of that technique, namely chaining [Everitt 1993]. This concept, though difficult to define formally, refers to the tendency of a technique to incorporate objects into existing clusters rather than to initiate new clusters.

2.1.4 Clumping Methods

Most clustering methods lead to distinct or disjoint clusters, but there are a number of techniques available which allow overlapping clusters. Such methods are often referred to as clumping techniques [Everitt 1993]. These techniques often begin with the calculation of a similarity matrix, followed by the division of the data into two groups by minimizing what is known as a *cohesion function*. Algorithms to minimize these functions proceed by successive reallocations of single individuals from an initial randomly chosen cluster center. By iterating from different starting points, many divisions into two groups may be found. In each case the members of the smaller group are noted and constitute a class to be set aside for further examination.

As discussed above, traditional clustering approaches can be generally classified into two major groups: hierarchical and non-hierarchical (optimization and density and mode-seeking techniques) approaches. For hierarchical clustering approaches, they perform their tasks based on the similarity matrix. Calculating such matrix requires computing all the pair-wise similarity values and thus it consumes a large amount of computational cost especially in a very large data set. It is, therefore, quite prohibitive for data mining applications. As opposed to hierarchical clustering approaches, non-hierarchical ones are computationally more efficient than hierarchical ones. However, both hierarchical and non-hierarchical approaches require to calculate the similarity between objects be determined based on the distance function that is defined in Euclidean space. They are thus not very effective in handling both continuous- and discrete-valued data. This results in the incapability to deal with many real-life databases containing both transaction and relational data for data mining applications. In addition, the use of distance function as similarity measure also leads to the problem in handling noisy and missing values in data. Besides these weaknesses, traditional clustering approaches are unable to provide descriptions characterizing clusters.

2.2 The artificial intelligence approaches

With most of the traditional clustering approaches, distance measures are required to perform their tasks. Such measures, as described before, always assume that the given data are continuous-valued data. It is, therefore, not well applicable to discrete-valued data. Because of the need to handle such data, conceptual clustering techniques [Michalski 1980; Fisher, Pazzani and Langley 1991; Fisher, Pazzani and Langley 1991; Fisher 1987; Fisher *et al.* 1993; Anderson and Matessa 1991; Gennari, Langley and Fisher 1990; and Cheesman and Stutz 1995] developed by the machine learning researchers can also be considered for the clustering of records containing discrete-valued data. Some of these techniques do not require the definition of a distance measure and can therefore be used to handle discrete-valued attributes. For example, CLUSTER/2 [Michalski 1980] forms categories those have good conjunctive expressions of features that are common to all or most category members. The algorithm generates a disjoint hierarchy of concepts by iteratively optimizing against two criteria of goodness: 'simplicity' and 'fit' so that preferences are given both to short conjunctive expressions (for the sake of comprehensibility) and to detailed conjunctive descriptions (for more characteristics about category members to be described and conveyed) [Fisher, Pazzani and Langley 1991]. However, CLUSTER/2 relies on several user-specified thresholds and background knowledge, such as problem constraints, properties of attributes, etc. to control its search.

UNIMEN [Lebowitz 1987] builds a classification tree based on a Hamming distance measure which mainly focuses on intra cluster similarity. WITT [Hanson and Bauer 1989] defines intra cluster similarity in terms of the strength of pairwise attribute relationships (co-occurrences) that exist within a cluster or group. The strength of these relationships is defined in terms of correlational measures that are represented as contingency tables.

Another well-known conceptual clustering technique is COBWEB [Fisher, Pazzani and Langley 1991; Fisher 1987; Fisher *et al.* 1993; Anderson and Matessa 1991; and Gennari, Langley and Fisher 1990]. It makes use of a category utility measure defined in terms of a posteriori probabilities to construct a classification tree. By maximizing the category utility, an optimal tree can be constructed for accurate prediction of attribute values. Clustering using COBWEB is incremental thus new objects can be assimilated into the existing hierarchy. Because of this, different data orderings can yield very different object groupings [Fisher *et al.* 1993; Anderson and Matessa 1991; and Gennari, Langley and Fisher 1990]. Even though some attempts [Biswas, Weinberg and Fisher 1998] have been made to overcome the problem by using merging and splitting operators, the problem cannot be completely eliminated. The measure in category utility also shows a tendency to bias toward larger, more inclusive categories.

In addition to the above, the clustering algorithm, AUTOCLASS, has been popular among data mining researchers [Cheesman and Stutz 1995]. It uses a fundamental finite mixture model to find an interclass mixture probability model that gives the chance of an instance belonging to different classes. Depending on the type of variables involved, AUTOCLASS makes different assumptions about their distributions [Cheesman *et al.* 1988; and Cheesman and Stutz 1995]. With these assumptions, groupings of objects are derived to maximize the posterior probability of individual clusters given the feature distribution. This maximization process allows the most probable number of classes to emerge from the data.

There are also some clustering algorithms related to the handling of discrete-valued data. For example, Cheng and Fu [Cheng and Fu 1985] developed HUATUO which produced intermediate conceptual structures for rule-based systems. Ralambondrainy [Ralambondrainy 1995] presented a conceptual version of the *k*-means algorithm for numeric and discrete data based on coding symbolic data numerically and using a mix of Euclidean and Chi-square distances to calculate the distance between the hybrid types of data that are represented using predicates as a group of attribute-value tuples joined by logical operators.

Many of the clustering algorithms in AI approach are able to handle discrete-valued data and some of them are also able to handle both continuous- and discrete-valued data. It is because AI researchers developed these methods for conceptual clustering [Stepp 1987]. It differs from traditional clustering approaches in such a way that not only does it discover the grouping of records, but also it determines a concept for each grouping discovered by clustering. Because of this, some algorithms perform their tasks with the need of domain knowledge. Even algorithms such as COBWEB and AUTOCLASS do not have such requirement, there is an ordering effect in COBWEB whereas the searching procedure is endless in AUTOCLASS [Cheesman and Stutz 1995]. In addition, all of them are unable to discover overlapping clusters.

2.3 Neural Network approaches

In the past few years, neural network approaches, methods which are model-based, have been used to solve a wide range of problems related to data mining such as classification. On one hand, to deal with classification problems, supervised training approaches such as traditional backpropagation neural network are frequently used. On the other hand, the unsupervised training approaches in neural network can be applied to solve the clustering problems. Actually, unsupervised learning is roughly equivalent to "clustering analysis" and does not require a representative for unsupervised learning [Diederich and Joachim 1990]. In competitive learning, a technique which is most representative for unsupervised learning, the procedure classifies a set of input vectors into disjoint clusters, in a way that elements in a cluster are similar to each other. It is called competitive learning because there is a set of hidden units that competes with each other to become active and to do the weight change. The winning unit increases its weights on input-lines with high input and decreases weights on links with small input. Since there is a constraint to each weight vector that the sum of weights has to be constant, the winning hidden unit will be selective to the input-vector where it was active, and not to other input patterns.

In competitive learning, perhaps the well-known neural network approach for clustering is the self-orgainizing map (SOM) which is proposed by Kohonen. Kohonen's self-organizing fearure map [Dayhoff 1990; Kohonen 1984; and Kohonen 1997] is a two-layered network that can organize a topological map from a random starting point. The resulting map shows the natural relationships among the patterns that are given to the network. The network combines an input layer with a competitive layer of processing units, and is trained by unsupervised learning.

The Kohonen feature map finds the organization of relationships among patterns. Incoming patterns are classified by the units that they activate in the competitive layer. Similarities among patterns are mapped into closeness relationships on the competitive layer grid. After training is completed, pattern relationships and groupings are observed from the competitive layer. The Kohonen network provides advantages over classical pattern-recognition techniques because it utilizes the parallel architecture of a neural network and provides a graphical organization of pattern relationships.

Even techniques in neural network approach are very powerful modeling tool to discover relationships in data, the discovered cluster is relatively difficult to understand [Fayyad, Piatetsky-Shapiro, and Smyth 1996b]. It is mainly due to their black-box characteristic of these techniques. Also, it is known that the computational complexity for neural network is quite high. It requires exhaustive training time for adjusting the weights to convergence below a certain threshold. This may result in computationally prohibitive for large amount of data in data mining applications. In addition, neural network techniques do not allow missing input values and they do not do very well in handling noisy data. Because the optimization function inside a neural network is often defined in Euclidean space, it is not clear how they are able to handle both continuous- and discrete-valued data. Furthermore, they are not designed to discover overlapping clusters.

2.4 The Data Mining approaches

Recently, the subject of clustering has been widely studied for spatial data mining. A number of effective algorithms have been developed for such tasks [Ng and Han 1994; Ester, Kriegel, Sander, Xu 1996; and Zhang, Ramakrishnan, and Livny 1996]. These algorithms share many characteristics in common. In particular, almost all of them perform their tasks by optimizing against a certain criteria. For example, CLARANS [Ng and Han 1994], applies a random search–based method to find an optimal clustering through minimizing a cost function defined as the distance between a representative object of each cluster and other objects. DBSCAN [Ester *et al.* 1996] finds clusters of points that are above a certain density. Such density is defined as the number of points that are inside a user-specific distance. BIRCH [Zhang, Ramakrishnan, and Livny 1996] uses a distance measure on the point objects to be clustered and forms groupings that minimize some measurements. In all cases, it is assumed that the distance measures are defined in the Euclidean space. For this reason, these algorithms are used mainly to cluster spatial data. In the case of discrete-valued data for which Euclidean distances cannot be defined, these techniques cannot be applied.

For tasks that involve discrete-valued data, some clustering techniques have been developed [Michaud 1997; Huang 1997a; Huang 1997b; Huang 1998; Li and Biswas 1997; and Ramkumar and Swami 1998; Han *et al.* 1998; Ganti, Gehrke and Ramakrishnan 1999; and Gibson, Kleinberg and Raghavan 2000]. Condorcet [Michaud 1997], for example, is one of them. It performs clustering by determining the similarity of two records using a simple voting principle. Two records are considered similar or dissimilar according to the number of positive and negative votes they receive. A positive vote is recorded if the corresponding values of an attribute are the same; otherwise, a negative vote is given. Once the similarity of a record is compared against all other records, clusters can be formed. A weakness of Condorcet is that the order of presentation of a record to the system may affect the clustering results. Furthermore, it does not explain the clustering results and is unable to handle incomplete and noisy data well.

Similar to Condorcet, a *k*-prototypes algorithm [Huang 1997a], which can also handle discrete-valued data, has also been proposed to solve the clustering problem in data mining. This algorithm can dynamically update *k* prototypes, corresponding to *k* clusters, to maximize an intra-cluster object similarity measure. This measure is defined as the weighted sum of both an Euclidean distance measure (for continuous-valued attributes) and a Hamming distance measure (for discrete-valued attributes). Unfortunately, the adding up of two distance measures defined in two different measurement spaces is not very meaningful. Furthermore, like Condorcet, it also suffers from the same problem of not being able to handle noisy and incomplete data well. In addition to *k*-prototypes algorithm, an algorithm similar to *k*-means named *k*-modes algorithm has also been proposed [Huang 1997b; and Huang 1998]. The algorithm suggests replacing the Euclidean distance for continuous-valued data to Hamming or Chi-square distance for discrete-valued data. This shares the advantage of the efficiency of *k*-means algorithm in large data set.

Because of the beauty of *k*-means algorithm in the aspect of efficiency, many data mining researchers aim at improving it for knowledge discovery process. For example, [Bradley, Fayyad 1998; and Fayyad, Reina, and Bradley 1998] refines the method for initialization of cluster center. Traditionally, the cluster center initialization in *k*-means is in a random manner. It is possible that the result of clustering is trapped in local optima due to the poor initial cluster centers. The algorithm makes use of the theory in expectation minimization to find a better cluster centers during initialization.

In addition to the above, the SBAC [Li and Biswas 1997] has also been proposed to handle clustering in data mining problems. It is a hierarchical clustering algorithm based on the well-known biological taxonomy and agglomerative technique. However, like the *k*-prototypes algorithm above, this approach also handles continuous- and discrete-valued data separately and requires different distance measure to be defined. In addition, due to the need to compute the distances between all pairs of objects, this approach can be computationally prohibitive for large volume of data.

In the past few years, owing to the success of frequent item sets in association, the frequent item sets are also used to apply on clustering. For example, in [Ramkumar and Swami 1998; Han *et al.* 1998; Ganti, Gehrke and Ramakrishnan 1999; and Gibson, Kleinberg and Raghavan 2000], the extraction of frequent item sets is employed. Although the method is simple and easy to understand, the process to obtain the frequent item sets depends on the *support* and *confidence* values that are determined in a subjective manner. In addition to the subjective determination of parameter values, such methods may not be used for continuous-valued attributes as the original idea of the frequent item sets is used in transaction data.

As a short summary, many of the existing data mining clustering approaches are designed for handling continuous-valued data in spatial databases. These approaches are very efficient in handling such data whose characteristics are large data set size and small number of dimensions. With such characteristics, they may not be very good when data to be dealt with contain both continuous and discrete values or when the number of dimensions is large in typical relational databases. Because of these reasons, some algorithms are developed to handle such cases. However, these algorithms either become computationally expensive or employ separate distance measure for different type of data. Some of them even require providing some subjective input parameters for their algorithms. In addition, for those algorithms using distance function for similarity, they are also unable to handle noisy or missing values in data. It should be noted that all of the existing clustering algorithms are unable to discover overlapping clusters and do not provide descriptions characterizing clusters. Therefore, the discovered knowledge cannot be represented explicitly.

Chapter 3

An Effective Algorithm for Clustering

To effectively perform clustering for data mining, we propose a new clustering algorithm in this chapter. Our algorithm has the following characteristics. First, unlike many existing clustering techniques which perform their tasks by taking a hierarchical approach [Anderberg 1973; Jain and Dubes 1988; and Everitt 1993], our algorithm employs a non-hierarchical approach so that some well known drawbacks such as updating of membership matrix, complexity in computing of distance functions during each iteration, overall complexity of the algorithms, and inability to determine cluster memberships for some objects or patterns, etc. [Everitt 1993] in hierarchical approach are avoided. Second, our algorithm employs a GA to search for an optimal grouping of records. During the evolutionary process, we define an explicit fitness function to guide the search. Third, the defined fitness measure is derived from a probabilistic similarity measure making it possible for it to be defined even when the data being dealt with contain noisy, dynamic, incomplete, missing, or even erroneous values. Fourth, such measure can also enable us to discover overlapping clusters. Fifth, our clustering algorithm is also able to extract interesting patterns in an objective manner. Therefore, it can perform its tasks without the need of any user-supplied input or domain knowledge about the problem. Sixth, a set of rules is discovered to characterize the specific grouping encoded in a chromosome. This makes the patterns underlying the database to be explicit.

In the next section, we present a formal description of our clustering problem for data mining task. In section 3.2, a description of a simple GA is given. After that, a novel method for clustering using a genetic algorithm (GA) is presented in Section 3.3. With the powerful evolutionary process using GA, we propose a fitness function based on the *support* and *confidence* values with an objective measure in the same section. Finally, in Section 3.4, we give the summary remarks of our algorithm and some of the existing clustering algorithms in the aspect of computational complexity and number of clusters.

3.1 Notations

The proposed clustering algorithm is described in the following using the following notations. Given a database, D containing N records, $R_1, \ldots, R_q, \ldots, R_N$, with $q = 1, 2, \ldots, N$. Suppose that each record in the database is described by n distinct attributes, $A_1, \ldots, A_j, \ldots, A_n$, with respective domains of $dom(A_1), \ldots, dom(A_j), \ldots, dom(A_n)$, A_j can take on continuous, or discrete data values. Suppose also that the data are noisy in the sense that a certain percentage of values are incomplete, inconsistent or incorrect so the patterns inherent in the data are not completely deterministic. The problem we are concerned with is to discover the meaningful grouping from the given database, D into clusters without any a priori knowledge about the data and without the need to assume that $\bigcup_{k=1}^{K} C_k = \{R_1, R_2, \ldots, R_n\}$ and $C_k \cap C_l = \emptyset, k \neq l$ where $C_k = \{R_q \mid q \in \{1, 2, \ldots, n\}\}$ and $k = 1, 2, \ldots, K$, K is the total number of discovered clusters. We assume that a record can belong to one or more clusters.

3.2 Genetic Algorithm

As our clustering algorithm [Chan and Chung 1999] is based on the use of a genetic algorithm (GA), we begin with a description of it. GA was first invented to mimic some of the processes observed in natural evolution [Davis 1991, Michalewicz 1996, DeJong 1988 and Goldberg 1989] and has been widely used for various optimization problems [Davis 1991]. The features of natural evolution were incorporated in a computer algorithm to yield a technique for solving difficult problems in the way that nature has done, i.e. through evolution. A GA carries out simulated evolution on populations of chromosomes which are actually strings of 1's and 0's. The simplest GA carries out simulated evolution on populations of such chromosomes. Like nature, the GA solves the problem of finding good chromosomes by manipulating the material in the chromosomes blindly through combining and recombining to reproduce high-quality combinations. The only information that a GA makes use of is an evaluation of each chromosome produced so that chromosomes with better evaluations are reproduced more often than those with bad evaluations [Davis 1991, Michalewicz 1996, DeJong 1988; and Goldberg 1989]. Since GA uses population-wide instead of point-search and the transition rules it employs are stochastic rather than deterministic, the probability of GA reaching a false peak during a search process is therefore much smaller than that with other conventional optimization methods. Although GAs cannot guarantee that the global optimum is always found, they can guarantee that a solution that is at least as good as the known ones can always be identified. An added appealing feature of GA is that it can be implemented on parallel computer architecture and can be made computationally very efficient [Wu *et al.* 1998].

In brief, a typical GA works according to two major mechanisms: (i) encoding which encodes the solutions to a problem in a chromosome and (ii) evaluation which determines how fit a chromosome (how good the solution it encodes) in the context of the problem is. With these two mechanisms, a GA searches for the best solution in the solution space by using some genetic operators (crossover and mutation) to manipulate genetic materials. All of these characteristics construct a powerful scheme to search for a better solution. A typical GA has several steps as follows:

Step 1: Create an initial generation consisting of a number of different individuals.

Step 2: Evaluate the fitness of each individual in the initial generation.

Step 3: Create a number of new individuals based on the current generation of individuals using the operators of selection, crossover, and mutation.

Step 4: Discard existing individuals to make room for the newly created ones.

Step 5: Evaluate the fitness of each newly created individual.

Step 6: If the termination condition is not met, go back to step 3.

Step 7: Return the best individual(s).

A simple GA typically uses a generational replacement technique. This technique replaces the entire set of population from parents to children during reproduction process. However, it has two potential drawbacks: (i) many of the best individuals inside the parent population may not reproduce at all and thus their genes may then be lost, (ii) the best genes may also be altered or destroyed after mutation or crossover process. To overcome these potential drawbacks, one solution is to modify the reproduction technique so that it replaces only one or two individuals at a time rather than all the individuals in the population [Davis 1991, Whitley and Kauth 1988; and DeJong 1988]. Genetic Algorithm that adopts this reproduction strategy has been referred to as steady state genetic algorithm (SSGA) and our clustering algorithm is based on it. Briefly, it has the following characteristics:

i) it creates one or two children through reproduction;

ii) it deletes one or two least fit members of the population to make room for the children, and

iii) it evaluates and then inserts one or two fittest children into the population.

Although a simple GA is easily implemented and less overhead, it produces a lot of duplicate chromosomes during evolutionary process. This may weaken the evolution power of the genetic algorithm. To overcome such drawback, we adopt steady state without duplicates to discard children that are duplicates of current chromosomes in the population rather than inserting them into the population. Fig. 2 describes the details of the SSGA algorithm.

```
population SSGA($P_{t-1}$)
begin
        $Chromosome_1 = select\ (P_{t-1})$;
        $Chromosome_2 = select\ (P_{t-1})$;
        crossover ($Chromosome_1, Chromosome_2$);
        mutate ($Chromosome_1$);
        mutate ($Chromosome_2$);
        $P_t = steady\text{-}state\ (P_{t-1}, Chromosome_1, Chromosome_2)$;
        return $P_t$;
end
```

Figure 2. The SSGA algorithm.

In brief, GAs are a randomized search method, they can provide stochastic optimization based on the mechanism of natural selection and natural genetics [Krishna and Murty 1999]. Also, they perform parallel search in complex search spaces. For this reason, there are some studies in applying GA on clustering tasks. Among existing clustering algorithms using GA [Tseng and Yang 1997; KunCheva and Bezdek 1998; Bezdek *et al.* 1994; Su and Chang 1998; Krishna and Murty 1999; Jiang and Ma 1996; Song *et al.* 1997; and Kemenade 1996], they mainly optimize for the followings: (i) Finding proper number of clusters [Tseng and Yang 1997]; (ii) Initializing cluster centers [KunCheva and Bezdek 1998; Bezdek *et al.* 1994]; and (iii) Finding optimal partition of data globally [Su and Chang 1998; Krishna and Murty 1999; Jiang and Ma 1996; Song *et al.* 1997; Kemenade 1996; KunCheva and Bezdek 1998; Bezdek *et al.* 1994]. No matter which of the above items they optimize for or which methods they are based on (e.g. *k*-means or Kohonen network), these algorithms employ a function evaluating the similarity based on a distance measure. Such measure, as explained before, is designed for handling continuous-valued data only. With both continuous- and discrete-valued data commonly found in real-life database systems, these algorithms are unable to perform their tasks well.

3.3 The Proposed Clustering Algorithm

To make use of GA to deal with the problem we are concerned with, we need the followings: (i) an encoding scheme and an initialization approach; (ii) a set of genetic operators including parent selection, crossover and mutation to manipulate the chromsomes; (iii) the fitness function; and (iv) the termination of evolutionary process. In this section, we give a detailed description of our algorithm.

3.3.1 Encoding Scheme and Initialization

To find the best grouping of records, each possible cluster arrangement is encoded in a single chromosome so that the q^{th} gene in the chromosome contains a cluster label, C_k that R_q belongs to. In other words, a specific grouping is represented in a chromosome of length N as shown in Fig. 3. Unlike some GAs which use binary symbols, our algorithm uses a symbolic representation scheme for its chromosome. During the initialization of the first population, each cluster label is represented in symbolic form and randomly assigned to the gene of each chromosome. Fig. 4 shows the details of the initialization of chromosomes. However, in cases where it is possible. the results of other clustering algorithms such as AUTOCLASS [Cheesman and Stutz 1995], COBWEB [Fisher 1987], Condorcet [Michaud 1997] and the *k*-modes [Huang 1997b] algorithms could be incorporated into the initial population. This will be able to ensure that the grouping of records discovered will at least be as good as the existing best. Also, it should be noted that there is a checking of duplication of chromosomes during intialization. For instance, a simple "re-numbering" of clusters representing the same clustering solution would be eliminated and a new chromosome would be re-generated instead.

	R_1	R_2	…	…	R_{N-1}	R_N
Cluster:	C_1	C_2	…	…	C_1	C_3

Figure 3. The genes of a chromosome

```
population initialize()
Let    C be a set of cluster label,
       popsize be the population size,
       N be the number of record in S.
begin
       $i = 1$;
       while ($i \le popsize$) do
             for ($j = 1; j \le N; j{+}{+}$) do
                   $chromosome_i.allele_j = rand(C)$;
             end for
             for ($k = 1$; $k < i$; $k{+}{+}$) do
                   if not $duplicate(chromosome_i, chromosome_k)$ then
                         $i{+}{+}$;
                   end if
             end for
       end for
       return $\bigcup_i chromosome_i$;
end
```

Figure 4. The initialization function.

3.3.2 The genetic operators

In the evolutionary process, several parameters and genetic operators have to be defined. In this section, we describe the parent selection, crossover and mutation methods for our proposed algorithm. In particular, the reasons why the proposed genetic operators are used are also given.

3.3.2.1 Parent Selection

The purpose of parent selection in a genetic algorithm is to give more reproductive chances, on the whole, to those population members that are the most fit. One commonly used technique is roulette wheel parent selection. It works as follows: (i) summing the fitness of all the population members; call the result total fitness. (ii) generating *n*, a random number between 0 and total fitness. (iii) returning the first population member whose fitness, added to the fitness of the preceding population members, is greater than or equal to *n*.

The effect of roulette wheel parent selection is to return a randomly selected parent. Although this selection procedure is random, each parent's chance of being selected is directly proportional to its fitness. On balance, over a number of generations this algorithm will drive out the least fit population members. This parent selection technique has the advantage that it directly promotes reproduction of the fittest population members by biasing each member's chances of selection in accord with its evaluation.

3.3.2.2 Crossover

Among the genetic operators, one-point crossover and mutation are the well-known operators for the evolutionary process. However, they have some drawbacks. No matter where the crossover point is selected, the gene at the head and tail of a chromosome must be broken up. This may potentially reduce the evolution power of the crossover operation. Therefore, uniform list crossover [Davis 1991, Whitley and Kauth 1988, and DeJong 1988] is adopted. It differs from one-point crossover in such a way that the location of the encoding of a feature on a chromosome is irrelevant to uniform crossover. It works as follows. Two parents are selected, and two children are produced. For each gene position on the two children, we decide randomly which parent contributes its gene value to which child [Davis 1991, Michalewicz 1996; DeJong 1988; and Goldberg 1989]. Fig. 5 shows uniform crossover in action. For each gene position on the parents, the template indicates which parent will contribute its value in that position to the first child. The second child receives the gene value in that position from the other parent. The uniform crossover technique is best suited with our problem because the set of records given is also unordered. Fig. 6 shows the detailed crossover algorithm:

Parent 1	C_1	C_3	C_2	C_1	C_4	C_3	C_1
Parent 2	C_3	C_1	C_2	C_1	C_1	C_4	C_2
Template	1	0	0	1	0	1	1
				Yields			
Child 1	C_1	C_1	C_2	C_1	C_1	C_3	C_1
Child 2	C_3	C_3	C_2	C_1	C_4	C_4	C_2

Figure 5. An Example of Uniform Crossover

```
void crossover (chromosome_1, chromosome_2)
Let     N be the number of record in S.
        pcrossover be the preset threshold for crossover
begin
        for (i = 1; i ≤ N; i++) do
                if random() < pcrossover then
                        temp = chromosome_1.allele_i;
                        chromosome_1.allele_i = chromosome_2.allele_i;
                        chromosome_2.allele_i = temp;
                end if
        end for
end
```

Figure 6. The crossover function.

3.3.2.3 Mutation

As with crossover operator, uniform list concept is also applied to mutation operation. Fig. 7 shows the detailed mutation function for our algorithm.

```
void mutate (chromosome)
Let     N be the number of record in S.
        pmutate be the preset threshold for crossover
        c be the set of cluster label.
begin
        for (i = 1; i ≤ N; i++) do
                if random() < pmutate then
                        chromosome.allele_i = rand(c);
                end if
        end for
end
```

Figure 7. The mutate function

With these uniform crossover and mutation operators together with the steady state without duplicates evolutionary scheme, a powerful searching strategy is thus constructed for the hill-climbing process.

3.3.3 The fitness function

To determine how good the cluster arrangement in a chromosome is, we propose a fitness function [Chung and Chan] here. This fitness function is defined in terms of a probabilistic similarity which is defined in turn by the *support* and *confidence* measures [Agrawal, Imielinski, and Swami 1993, Agrawal, Srikant 1994, and Agrawal *et al.* 1996]. With this function, the fitness of each chromosome is evaluated in two steps: (i) detection of important attribute-value pairs underlying the record grouping encoded in a chromosome for the construction of interesting rules, and (ii) determination of chromosome fitness by matching of discovered rules. In the following, we discuss each of these steps in details.

3.3.3.1 Detection of interesting rules

The aim of this step is to discover interesting rules that are useful for the characterization of the clusters encoded in a chromosome. To do so, it should be noted that a record R in a relational database is made up of a set of attribute-value pairs, i.e. A_i-V_{ij} pairs. We define I as a set of attribute-value pairs in R. The support of a set of attribute-value pairs, support(I) and the support of a cluster label, support(C_k) are therefore defined as the probability that these attribute values exist in database and the probability that a cluster label exists in database respectively:

$$\text{support}(I) = \frac{\text{no. of records that } I \text{ occur}}{M} = \Pr(I) \tag{4}$$

$$\text{support}(C_k) = \frac{\text{no. of records in } C_k}{M} = \Pr(C_k) \tag{5}$$

where M is the total number of records counted. The value of M should be equal to or less than N due to the possibility of having missing values in the database.

Similarly, the support of records that are characterized by I and belong to C_k is defined as:

$$\text{support}(IC_k) = \frac{\text{no. of records in } C_k \text{ that is characterized by } I}{M}$$

$$= \Pr(I \cap C_k) \tag{6}$$

To determine how a set of attribute-value pairs is actually correlated to a cluster label, C_k, we find the confidence which is defined as:

$$\text{confidence}(I \Rightarrow C_k) = \frac{\text{support}(IC_k)}{\text{support}(I)} = \Pr(C_k \mid I) \tag{7}$$

In other words, the confidence value reflects how much confidence that a record characterized by I belongs to cluster label C_k. The higher the confidence value, the greater the number of records that we can find I also in C_k.

To know if a set of attribute-value pairs, say I, is the important attribute values for a cluster label, say C_k, we determine if confidence($I \Rightarrow C_k$) is significantly different from support(C_k). If this is the case, then I can be considered as the important attribute values.

To determine how great the difference between confidence($I \Rightarrow C_k$) and support(C_k) (i.e. $\Pr(C_k \mid I)$ and $\Pr(C_k)$) is considered to be significant, we propose a method here. Let o_{IC_k} be the total number of records in C_k and also with the characteristic I and, e_{IC_k} be the expected value of o_{IC_k}, then $o_{IC_k} = \text{support}(IC_k) \times M$ and $e_{IC_k} = \text{support}(I) \times \text{support}(C_k) \times M$. The adjusted residual [Chan, Wong and Chiu 1988, Chan and Wong 1990], d_{IC_k} is:

$$d_{IC_k} = \frac{z_{IC_k}}{\sqrt{v_{IC_k}}} \tag{8}$$

where $z_{IC_k} = \dfrac{o_{IC_k} - e_{IC_k}}{\sqrt{e_{IC_k}}}$, and (9)

$$v_{IC_k} = [(1 - \text{support}(I))][(1 - \text{suppot}(C_k))] \tag{10}$$

There are two components in the adjusted residual. The first component is the standardized residual, z_{IC_k} [Chan, Wong and Chiu 1988, Chan and Wong 1990]. It has an approximate normal distribution with a mean of approximately 0 and a variance of approximately 1. The second component v_{IC_k} is the maximum likelihood estimate of the variance of z_{IC_k}. It is calculated from support(I) and support(C_k). A d_{IC_k} provides a better approximation to a standard normal variable than z_{IC_k} and shares almost the same properties in z_{IC_k}.

With this measure, we can now consider how significant the difference between confidence($I \Rightarrow C_k$) and support(C_k) by comparing the absolute value of d_{IC_k} with 1.96, the 95 percentiles of the normal distribution (or higher for a greater confidence level). If $|d_{IC_k}| \geq 1.96$, we can conclude that the discrepancy between confidence($I \Rightarrow C_k$) and support(C_k) is significantly different and I is the important attribute values to the arrangement of cluster label. The sign of d_{IC_k} also tells whether the presence or the absence of I is an important pattern for C_k. A $d_{IC_k} >$ +1.96 indicates that the presence of I is a relevant pattern for C_k. In other words, given that a record is characterized by I, it is more likely for it to be a member of C_k than other clusters. Similarly, if $d_{IC_k} < -1.96$, it indicates the absence of I is a relevant pattern for C_k. On the other hand, if $|d_{IC_k}| < 1.96$, it indicates that the presence of I is not an important pattern for C_k. Records characterized by I therefore show no correlation with any cluster label and yield no information to the arrangement of cluster label. Such attribute-value pairs are irrelevant for the clustering process and hence they are discarded from further analysis.

Once the important attribute values are extracted through the adjusted residual, these patterns can be explicitly represented in a form of if-then rules:

IF <*condition*> THEN <*conclusion*> with *support* and *confidence*.

Where the *condition* part specifies a set of attribute-value pairs and the *conclusion* part specifies a cluster label. The *support* and *confidence* values associated with the rule indicate the degree of support and confidence of a record having the specific I and cluster label C_k. For those if-then rules that have a positive sign of d_{IC_k}, we can also interpret them as the positive association between I and the corresponding cluster C_k. Similarly, for those if-then rules that have a negative sign of d_{IC_k} we can then interpret them as the negative association for C_k. Having these positive and negative associations determined from an objective measure to describe the existing grouping, data interpretation of such grouping is achieved at the same time when detection of pattern is taken place.

3.3.3.2 Measuring fitness by matching interesting rules

Once the interesting rules with the values of *support* and *confidence* are constructed, the fitness of chromosomes can be evaluated through these interesting rules. This is performed by searching through the rules to determine which interesting attribute-value pairs of a record are important in determining which cluster such record belongs to. To quantify how attribute value pairs actually belong to a cluster, the weight of evidence, W associated with each interesting attribute-value pairs, I to the corresponding C_k is proposed. It measures the amount of positive and negative relationships supporting or refuting a record to be clustered in C_k. It is defined as:

$$W(\text{Cluster} = C_k / \text{Cluster} \neq C_k \mid I) = MI(\text{Cluster} = C_k \mid I) - MI(\text{Cluster} \neq C_k \mid I) \quad (11)$$

$$\text{where} \quad MI(\text{Cluster} = C_k : I) = \log \frac{\Pr(\text{Cluster} = C_k \mid I)}{\Pr(\text{Cluster} = C_k)} \quad \text{and} \quad (12)$$

$$MI(\text{Cluster} \neq C_k : I) = \log \frac{\Pr(\text{Cluster} \neq C_k \mid I)}{\Pr(\text{Cluster} \neq C_k)} \quad (13)$$

It should be noted that equation (12) and (13) can also be re-written as follows:

$$MI(\text{Cluster} = C_k : I) = \log \frac{\text{confidence}(IC_k)}{\text{support}(C_k)} \quad (14)$$

$$MI(\text{Cluster} \neq C_k : I) = \log \frac{\text{confidence}(I\overline{C_k})}{\text{support}(\overline{C_k})} \tag{15}$$

In brief, the derivation of weight of evidence, W is based on an information theoretic measure derived from the mutual information. It is composed of two components: $MI(\text{Cluster} = C_k : I)$ and $MI(\text{Cluster} \neq C_k : I)$ which indicate the positive and negative evidence that supports or refutes a record to be clustered into C_k given the interesting attribute-value pairs, I. From equation (12), $MI(\text{Cluster} = C_k : I)$ is positive if and only if $\Pr(\text{Cluster} = C_k \mid I) > \Pr(\text{Cluster} = C_k)$, otherwise it is either negative or has a value 0. This shows that the higher the value of $MI(\text{Cluster} = C_k : I)$, the more the record characterized by attribute-value pairs I is in favor of being clustered into C_k. Similarly, higher value of $MI(\text{Cluster} \neq C_k : I)$ indicates the record characterized I should not be assigned into C_k.

In other words, the weight of evidence may be interpreted as an overall measure of the difference in the gain of information when a record is assigned to C_k and when it is assigned to other clusters based on I. The weight is positive if I provides positive evidence supporting the assignment of a record to C_k; otherwise, it is negative. Noted also that W can also be expressed, equivalently, as:

$$\begin{aligned} W(\text{Cluster} = C_k / \text{Cluster} \neq C_k \mid I) &= \log \frac{\Pr(I \mid \text{Cluster} = C_k)}{\Pr(I \mid \text{Cluster} \neq C_k)} \\ &= \log \frac{\text{confidence}(C_k \Rightarrow I)}{\text{confidence}(\overline{C_k} \Rightarrow I)} \end{aligned} \tag{16}$$

It should be noted that $W(\text{Cluster} = C_k / \text{Cluster} \neq C_k \mid I)$ is unable to obtain when $\Pr(I\overline{C_k}) = 0$ and $\Pr(IC_k) = 0$. If $\Pr(I\overline{C_k}) = 0$, $W(\text{Cluster} = C_k / \text{Cluster} \neq C_k \mid I)$ is replaced by $\max_{k,p} W(\text{Cluster} = C_k / \text{Cluster} \neq C_k \mid I_p)$ given $\Pr(I_p\overline{C_k}) \neq 0, p = 1, 2, \ldots, P$ and P is the total number of all possible values of I. Similarly, if $\Pr(IC_k) = 0$,

$W(\text{Cluster} = C_k\,/\,\text{Cluster} \neq C_k \mid I)$ is replaced by $\min_{k,p} W(\text{Cluster} = C_k\,/\,\text{Cluster} \neq C_k \mid I_p)$ given $\Pr(I_p C_k) \neq 0, p = 1, 2, \ldots, P$ and P is the total number of all possible values of I.

With rules and weight of evidence, a record is considered to be matched with a rule if and only if both the *condition* and *conclusion* parts of the rule are matched with the values of the attributes and the cluster label reflected in the gene of chromosome respectively. In this case, the matched rule can now be considered as providing some information of the record, R_q to belong to a particular cluster. By repeating this search procedure to match each attribute value and cluster label of R_q in the chromosome against the rules, the degree of supporting or refuting R_q belonging to cluster label, C_k, can easily be determined. To quantitatively estimate and combine the evidences from these matched rules, a measure of evidence is needed. Suppose that, of the n attributes that describe R_q, only m $(m \leq n)$ of them, $\text{val}_{[1]}, \ldots, \text{val}_{[j]}, \ldots, \text{val}_{[m]}$ with $\text{val}_{[j]} \in \{\text{val}_j \mid j = 1, \ldots, n\}$, are found to match one or more rules, then the weight of evidence $W_{R_q C_k}$ of a record, R_q with corresponding cluster label, C_k can be defined as:

$$\begin{aligned} W_{R_q C_k} &= W(\text{Cluster} = C_k\,/\,\text{Cluster} \neq C_k \mid \text{val}_1, \ldots, \text{val}_n) \\ &= W(\text{Cluster} = C_k\,/\,\text{Cluster} \neq C_k \mid \text{val}_{[1]}, \ldots, \text{val}_{[m]}) \qquad (17) \end{aligned}$$

If there is no a priori knowledge concerning the interrelation of the attributes in the problem domain, then the weight of evidence provided by all the attribute values of R_q in favor of it being assigned to C_k as opposed to other clusters is equal to the sum of the weights of evidence provided by each individual attribute value of R_q that is relevant for the clustering task. That is:

$$W_{R_q C_k} = \sum_{j=1}^{n} W(\text{Cluster} = C_k\,/\,\text{Cluster} \neq C_k \mid \text{val}_{[j]}) \qquad (18)$$

Hence, intuitively, the total amount of evidence supporting or refuting a certain cluster assignment of R_q is equal to the sum of the individual pieces of evidence provided by each attribute value that characterizes R_q. After obtaining the measure supporting and refuting a record to be assigned to a cluster, the overall fitness of a chromosome is based on $W_{R_qC_k}$ and is defined as:

$$\text{fitness} = \sum_{k=1}^{K} \frac{\sum_{q=1}^{N} W_{R_qC_k}}{\text{support}(C_k) \cdot M} \tag{19}$$

According to equation (19), the fitness value of a chromosome is the summation of the average weight of all cluster labels. As the weight of evidence of a record reflects the goodness of the arrangement between a record and a cluster label, the higher weight indicates the better goodness. By maximizing the fitness, it is equivalent to maximize the average weight of evidence. The higher weight of evidence of a cluster shows that the more similar records are grouped together to that cluster and, at the same time, the more dissimilar these records from any other clusters. In other words, on maximizing the fitness value, intra-class similarity is maximized and inter-class similarity is minimized. The highest fitness chromosome therefore shows the best overall arrangement of the cluster labels and thus it is an optimal solution of the clustering problem.

Since $W_{R_qC_k}$ is an overall measure of the difference in the gain of information when R_q is assigned to C_k, the sign of $W_{R_qC_k}$ provides information about if R_q should be assigned to C_k. Using such information, we can determine if R_q belongs to one or more clusters. Given a record, R_q and cluster labels C_i and C_j with $i,j = 1, 2, \ldots, K$, $i \neq j$, if both $W_{R_qC_i}$ and $W_{R_qC_j}$ are greater than zero, R_q has positive evidence to be assigned to both C_i and C_j. With such measure, the boundaries of cluster C_i and C_j become overlapped and R_q is allowed to be a member of one or more clusters.

3.3.4 Termination of Evolutionary Process

As mentioned before, a typical genetic algorithm may generate a lot of duplicate chromosomes on convergence. Using steady-state genetic algorithm without duplicates can solve such problem. Because of no duplicate chromosomes to be existed inside the whole population, all chromosomes cannot converge to the same point. We, therefore, stop the evolutionary process when the population converges to a stable fitness value or when the variance of the population is within 1% of the fitness value of the fittest individual.

Chapter 4

Experimental Results and Discussions

In this chapter, we evaluate our clustering algorithm by comparing its performance against several clustering algorithms. We begin by describing: (i) *k*-means, a traditional clustering approach; (ii) *k*-modes algorithm, a clustering categorical data approach in data mining; (iii) Condorcet, a demographic clustering approach; (iv) COBWEB, an incremental machine learning approach for clustering; and (v) AUTOCLASS, a parametric method based on Bayesian classification. In the following we give a description of these algorithms.

4.1 The clustering algorithms use for performance evaluation

There are currently many strategies for tackling the problem of clustering, namely, the *k*-means clustering algorithm, *k*-modes algorithm, Condorcet, COBWEB, and AUTOCLASS. The *k*-means clustering algorithm [Jain and Dubes 1988; Everitt 1993] is one of the famous clustering algorithms traditionally and is also applied to data mining community with a great success. *k*-modes [Huang 1997b] is a *k*-means like algorithm which is able to handle discrete-valued data and Condorcet [Michaud 1997] has been one of the popular algorithms in IBM's clustering approach for data mining. COBWEB and AUTOCLASS clustering approaches [Fisher 1987, Cheesman *et al.* 1988, and Cheesman and Stutz 1995] are originally proposed from the machine learning researchers. COBWEB has been widely used in conceptual clustering because of its incremental learning strategy. AUTOCLASS which is a parametric method based on Bayesian classification is also one of the well-known techniques in machine learning community. In this section, we describe all these algorithms to be used to compare with our clustering algorithm in the evaluation section.

4.1.1 K-means algorithm

In *k*-means clustering method and its variants, MacQueen's *k*-means [MacQueen 1967] is a well-known one. MacQueen's *k*-means clustering uses the term "*k*-means" to denote the process of assigning each data unit to that cluster (of k clusters) with the nearest centroid (mean). The key implication in this process is that the cluster centroid is computed on the basis of the cluster's current membership rather than its membership at the end of the last reallocation cycle as with the Forgy and Jancey methods [Everitt 1993]. MacQueen's algorithm for sorting m data units into k clusters is composed of the following steps: (1) Take the first k records in the data set as clusters of one member each. (2) Assign each of the remaining $m–k$ records to the cluster with the nearest centroid. After each assignment, re-compute the centroid of the gaining cluster. (3) After all records have been assigned in step 2, take the existing cluster centroids as fixed seed points and make one more pass through the data set assigning each record to the nearest seed point.

By using the first k records as seed points and relying on only one reallocation pass, this method achieves the distinction of being the least expensive of all clustering. The total effort from the initial configuration through to the final clusters involves only $k(2m\text{-}k)$ distance computations, $(k\text{-}1)(2m\text{-}k)$ distance comparisons, and $m\text{-}k$ centroid updates. This computational workload is only a small fraction of that involved in a hierarchical cluster analysis because k is usually much smaller than m. However, blindly using the first k records may be less than satisfactory unless the analyst can arrange to place his choices for the initial centroid at the front of the data set.

In order to reduce the effect on the choices for the initial centroid, the mentioned k-means algorithm is modified in such a way that an iterative pass of the algorithm is employed instead of a single pass [Everitt 1993]. Because of this, a convergent measure is required. A convergent clustering method using the k-means process can be implemented through the following sequence of steps. (1) Begin with an initial partition of the records into clusters. If desired, the partition could be constructed by using steps 1 and 2 of the ordinary MacQueen method. (2) Take record unit in sequence and compute the distance to all cluster centroids; if the nearest centroid is not that of the record's parent cluster, then reassign the record and update the centroids of the losing and gaining clusters. (3) Repeat step 2 until convergence is achieved; that is, continue until a full cycle through the data set fails to cause any changes in cluster membership.

4.1.2 *K*-modes algorithm

As mentioned, one of the major advantages of *k*-means algorithm is about its performance in large volume of data [Jain and Dubes 1988; Everitt 1993; and Huang 1997a, 1997b, 1998]. However, with the downside in handling discrete-valued data, *k*-modes algorithm [Huang 1997b] is proposed to improve such limitation. The *k*-modes algorithm has three major modifications to the *k*-means algorithm, i.e., using different dissimilarity measures, replacing *k* means with *k* modes, and using a frequency based method to update modes. For dissimilarity measures, *k*-modes algorithm measures the distance between two records defined by the total mismatches of the corresponding attribute categories of the two records (i.e., Hamming distance). The smaller the number of mismatches is, the more similar the two records. Since the use of Hamming distance may lead to many record pairs with equal distance, this may further result in the incapability to determine the dissimilarity between records. Therefore, besides using hamming distance, *k*-modes algorithm can also make use of the chi-square distance [Huang 1997b] as a dissimilarity measures.

The *k*-modes algorithm [Huang 1997b] consists of the following steps:

1. Select *k* initial modes, one for each cluster.
2. Allocate a record to the cluster whose mode is the nearest to it according to the distance measure. Update the mode of the cluster after each allocation according to the Theorem.
3. After all records have been allocated to clusters, retest the dissimilarity of records against the current modes. If a record is found such that its nearest mode belongs to another cluster rather than its current one, reallocate the record to that cluster and update the modes of both clusters.
4. Repeat step 3 until no record has changed clusters after a full cycle test of the whole data set.

Like the *k*-means algorithm, the *k*-modes algorithm also produces locally optimal solutions that are dependent on the initial modes and the order of records in the data set. There are therefore two methods proposed in *k*-modes algorithm for the initialization of modes. The first method selects the first *k* distinct records from the data set as the initial k modes. The second method to choose the highest frequency attribute value as the first mode in the sorted list.

4.1.3 Condorcet

Besides the k-modes algorithm [Hunag 1997b] which modifies the basic k-means algorithm to handle discrete-valued data in data mining applications, Condorcet [Michaud 1997] is another clustering algorithm for discrete-valued data. The basic concept of Condorcet is also based on distance measure. It measures the distance between input records and thus assigns these records to specific clusters. Pairs of records are compared by the values of the individual fields within them. The number of fields that have similar values determines the degree to which the records are judged to be similar. The number of fields that have dissimilar values determines the degree to which the records are judged to be different. This mechanism can be thought of as awarding scores for and against the similarity of the two records.

The distance measure employed by Condorcet is called the NCC measure. This NCC measures intraclass agreements as well as interclass disagreements and combines them in such a way that partitions that have small intraclass distances and large interclass distances will have higher measure. Such measure can be expressed as

$$F(Y) = \sum_{i=1}^{n} \sum_{j \neq i} (m - 2d_{ij}) y_{ij} \qquad (20)$$

where m be the total number of attributes and d_{ij} be the Hamming distance between two records i and j. And an equivalence relation using an $n \times n$ 0-1 matrix $Y = [y_{ij}]$, where $y_{ij} = 1$ if i and j belong to the same class and 0 otherwise.

From equation (20), the term $m - 2d_{ij}$ is actually the number of "agreements" minus the number of "disagreements" about record i and j being in the same class. With the NCC, the objective of performing clustering task is to maximize the NCC.

In other words, NCC makes use of a modified Hamming distance which is used to determine the similarity and dissimilarity between two records. When a pair of records has the same value for the same field, the field gets a vote of +1. When a pair of records does not have the same value for a field, the field gets a vote of –1.

The overall score is calculated as the sum of scores for and against placing the record in a given cluster.

A record is then assigned to another cluster if the overall score is higher than the overall scores if the record was assigned to any of the other clusters. If the overall scores turn out to be negative, the record is now a candidate for placing it in its own cluster. There are a number of passes over the set of records, and each record must be reviewed for potential reassignment to a different cluster. Clusters and their centers are therefore updated continuously within each pass, until either the maximum number of passes is achieved, or the maximum number of clusters is reached and the cluster centers do not change significantly as measured by a user-determined margin.

4.1.4 COBWEB

One of the difficulties for clustering discrete-valued data using traditional clustering methods, e.g. *k*-means [MacQueen 1967] and data mining methods, e.g. *k*-modes [Huang 1997b] and Condorcet [Michaud 1997] is about the distance measure which is always defined in the Euclidean space. To avoid such problem, COBWEB [Fisher 1987] is a conceptual clustering method which is not based on distance measure developed by machine learning researchers. It uses a heuristic measure called category utility to guide search. Gluck and Corter [Gluck and Corter 1985] originally developed this metric as a means of predicting the basic level in human classification hierarchies. Category utility can be viewed as a function that rewards traditional virtues held in clustering generally – similarity of records within the same class and dissimilarity of records in different classes [Fisher 1987]. In particular, category utility is a tradeoff between intra-class similarity and inter-class dissimilarity of records, where records are described in terms of discrete-valued attribute-value pairs. Intra-class similarity is reflected by conditional probabilities of the form $P(A_k = V_{ij} \mid C_k)$ where $A_i = V_{ij}$ is an attribute-value pair and C_k is a class. The larger this probability, the greater the proportion of class members sharing the value and the more predictable the value is of class members. Inter-class similarity is a function of $P(C_k \mid A_i = V_{ij})$. The larger this probability, the fewer the records in contrasting classes that share this value and the more predictive the value is of the class.

By combining the probabilities of individual values, an overall measure of partition quality can be evaluated for a set of mutually exclusive classes, $\{C_1, C_2, \ldots, C_n\}$. This is given by:

$$\sum_{k=1}^{n}\sum_{i}\sum_{j}P(A_i=V_{ij})P(C_k \mid A_i=V_{ij})P(A_i=V_{ij} \mid C_k)$$

$$=\sum_{k=1}^{n}P(C_k)\sum_{i}\sum_{j}P(A_i=V_{ij} \mid C_k)^2 \tag{21}$$

Finally, category utility can be defined as the increase in the expected number of attribute values that can be correctly guessed $(P(C_k)\sum_i\sum_j P(A_i=V_{ij} \mid C_k)^2)$ given a partition $\{C_1, C_2, \ldots, C_n\}$ over the expected number of correct guesses with no such knowledge $\sum_i\sum_j P(A_i=V_{ij} \mid C_k)^2$. Category Utility, CU($\{C_1, C_2, \ldots, C_n\}$) equals

$$\frac{\sum_{k=1}^{n}P(C_k)\sum_i\sum_j P(A_i=V_{ij} \mid C_k)^2-\sum_i\sum_j P(A_i=V_{ij})^2}{n} \tag{22}$$

The first term in the numerator measures the expected number of attribute-value pairs that can be guessed correctly by using the classification (within-class similarity). The second term measures the same quantity without using the classes (parent's within-class similarity). Category utility measures the increase of attribute-value pairs that can be guessed above the guess based on frequent alone. The measurement is normalized with respect to the number of classes.

When a new record is introduced, COBWEB tries to accommodate it into an existing hierarchy starting at the root. The system carries out one of the following operators determined by the category utility:

1. If the root has no subclasses, create a new class and attach the root and the new record as its subclasses;

2. Attach the record as a new subclass of the root;
3. Incorporate the record into one of the subclasses of the root;
4. Merge the two best subclasses and incorporating the new record into the merged subclass; or
5. Split the best subclass and again considering all the alternatives.

If the record is assimilated into an existing subclass, the process recurses with this class as the root of the hierarchy. Once again, COBWEB uses category utility to determine the next operator to apply.

Merging and splitting operators in COBWEB are used to guard against the effects of initially skewed data. To reduce unnecessarily merging and splitting, these operators are applied on the best hosts as indicated by category utility. These two operators are roughly inverse operators and allow COBWEB to move bi-directionally through a space of possible hierarchies. Splitting can be invoked to undo the effects of a prior merging should conditions change and vice versa. In general, merging is invoked when initial observations suggest that the environment is a space of highly similar objects, relative to the actual structure of the environment suggested by subsequent observations. Splitting is invoked when the environment is more 'compressed' than suggested by initial input. Merging and splitting decrease the sensitivity of COBWEB to input ordering due to their inverse relation.

4.1.5 AUTOCLASS

AUTOCLASS [Cheesman *et al.* 1988, and Cheesman and Stutz 1995] is a parametric approach for clustering. The fundamental model of AUTOCLASS is the classical finite mixture distribution. It gives the interclass mixture probability that an instance X_i is a member of class C_k, independently of anything else we may know of the instance. The interclass probability density function (p.d.f.) T_c is a Bernoulli distribution characterized by the class number K and the probabilities of the interclass parameter value instantiating a p.d.f., $\bar{V}_c$. Each class C_k is then modeled by a class p.d.f., giving the probability of observing the instance attribute values $\bar{X}_i$ conditional on the assumption that instance X_i belongs in class C_k. The class p.d.f. T_k is a product of individual or covariant attribute p.d.f.'s T_{kn}, $n=1,2,\ldots,N$ and N is the number of attributes; e.g. Bernoulli distributions for nominal attributes, Gaussian density densities for real numbers, Poisson distributions for number counts, etc. It is not necessary that the various T_k be identical, only that they all model the same subset of the instance attributes.

One of the major differences between AUTOCLASS to all other clustering algorithms is that it never assigns any instances to the classes. Instead of the assignment, AUTOCLASS makes use of a weighted assignment, which weights on the probability of class membership. The basic idea of AUTOCLASS holds that no finite amount of evidence can determine an instance's class membership and it further holds that the classification p.d.f. and parameter values constitute a more informative class description than any set of instance assignments. With this idea, the weighted assignment approach eliminates the brittle behavior that boundary surface instances can induce in classification systems that decide assignments [Cheesman and Stutz 1995]. In addition, it allows any user to apply decision rules appropriate to that user's current goals. Fig. 8 shows a high level description of the AUTOCLASS scheme.

1. Determine the number of groupings, k, for a partition.
2. Randomly split the data into k groups and estimate the parameters of each group.
3. Repeat the redistribution process using a hill climbing method in an attempt to optimize the partition structure, and
4. Adjust the number of k, and repeat steps 1-3.

Figure 8. A high level description of the AUTOCLASS scheme.

From Fig.8, the search method in AUTOCLASS is divided into two parts: (i) determining the number of classes; and (ii) determining the parameters that optimize the most probable class. To accomplish these two parts, AUTOCLASS begins with more classes and searches to find the best class parameters for that number of classes. Through approximating the integral, the relative probability of that number of classes is found. This process repeats with decreasing the number of classes until the most probable number of classes and parameters are discovered.

In determining the number of classes, AUTOCLASS suggests that including a class which has negligible posterior probability in the model cannot improve the likelihood of the data at all. Similarly, the model in which a class has negligible prior probability will always be less probable than models which simply omit that class. By setting a larger number of classes, the optimal classification is found when the resulting classes have significant probability and have empty classes once the number of classes is increased. In determining the parameters for defining each class, AUTOCLASS uses a Bayesian variant of Dempster and Laird's EM algorithm to find the best class parameters for a given number of classes.

4.2 Evaluation

In this section, we present the experimental results on our clustering algorithm. In section 4.2.1, a set of simulated data is generated to show in details how our algorithm performs its task. The set of simulated data contains 200 records and is characterized by 9 attributes. In section 4.2.2, several sets of real-life data are used to evaluate the performance of our algorithm. They were selected from a wide variety of applications such as medical, census, and scientific application, etc. The data size of these real-life data ranges from several hundreds to several tens of thousands records. Among the data sets, some of them contain only discrete-valued attributes, some of them contain only continuous-valued attributes and some of them contain both continuous- and discrete-valued attributes. For those data sets that contain missing values, the percentage of missing values in some attributes ranges from 0.5% to 97.5%.

To evaluate the performance of our method, we applied it to several real-life databases and compared the experimental results with that obtained using (i) COBWEB [Fisher 1987], a clustering algorithm developed by machine learning researcher for discrete-valued data; (ii) AUTOCLASS [Cheesman and Stutz 1995], a clustering algorithm also developed by machine learning researchers for both discrete- and continuous-valued data; (iii) *k*-means algorithm [Jain and Dubes 1988; Everitt 1993], a traditional clustering algorithm; (iv) *k*-modes algorithm [Huang 1997b]; a clustering algorithm developed by data mining researcher for discrete-valued data; and (v) Condorcet [Michaud 1997], a clustering algorithm also developed by data mining researcher for discrete-valued data.

In the experiment, we set the crossover rate and mutation rate to be 0.7 and 0.08 respectively for our algorithm. The number of chromosomes in population is 50. Also, ten sets of experiments were taken for each database.

For COBWEB, to reduce the chance of ordering effect which may reduce its effectiveness, ten sets of experiments were also taken. For each trial, the order of records is generated in a random manner. Out of the ten results, the highest accuracy is taken and reported.

The searching process in AUTOCLASS is endless [Cheesman and Stutz 1995]. It repeatedly creates a random classification and then tries to massage this into a high probability classification through local changes, until it converges to some "local maximum". It then remembers what it found and starts over again, continuing until the user tells it to stop. Each effort is called a "try". There are two basic methods to stop the AUTOCLASS searching: (1) a maximum number of "try" exceeded; and (2) a maximum duration passed. It is recommended that for moderately large to extremely large data sets (~200 to ~10,000 datum), it is necessary to run AUTOCLASS for at least 50 trials. In our experiments, to ensure a more optimal clustering result to be obtained, we set the number of trails equal 500. All the rest parameters used are the default setting.

For the *k*-means algorithm, cluster centers are initalized in a random manner. Transformation of all discrete-valued attributes to binary-valued attributes is taken so that *k*-means can handle both discrete- and continuous-valued data in experiments. For *k*-modes, a new cluster center initialization method proposed in [Huang 1997b] is employed. The maximum number of iterations is set to be 50 for Condorcet. As the same as COBWEB and our algorithm, ten sets of experiments were taken for *k*-means, *k*-modes and Condorcet algorithms and the highest accuracy is reported.

After describing the details of all algorithms, a measure is defined to evaluate the effectiveness of them. The misclassification counts measure in [Krovi 1992; Li and Biswas 1997 and Huang 1997b] is one of the typical evaluation methods. It computes the number of record misclassification in the partitional structures assuming the partitional structure presented in the base data as the true structure. The smaller this value, the closer the grouping discovered by the algorithm is to the real grouping. In our case, since data used in all the experiments presented in

Section 3.5.1 and 3.5.2 have known grouping, that is, the original classification of data is known, we can easily calculate the correct classification of all the evaluated algorithms and represent it as accuracy. The higher the accuracy, the better the performance of an algorithm is. The accuracy is given as:

$$accuracy = \frac{\text{total number of records in database - number of records misclassified}}{\text{total number of records in database}} \quad (23)$$

With the reason that some of the clustering algorithms such as COBWEB and Condorcet cannot pre-set the number of clusters at the beginning of experiments. The number of clusters discovered from these algorithms may not be equal to the number of classes in the original classification of databases. In this case, the accuracy of the algorithm is obtained by the following procedures: (i) if the number of clusters discovered by an algorithm is less than that in the original classification, match all clusters generated by the algorithm to the original classification so that the accuracy is the highest; otherwise, match all classes in the original classification to the clusters generated by the algorithm for the same purpose; (ii) for those unmatched classes in the original database or unmatched clusters generated by the algorithm, merge them one by one to the matched clusters in the previous step so that the accuracy is also the highest. In other words, the highest possible accuracy is recorded even when the number of clusters generated by a given algorithm is different from the original classification in the database.

4.2.1 Simulated Data

In this experiment, we generated a set of simulated data consisting of 200 records. Each record is characterised by 9 features of a house. The detailed description of the attributes and their values are tabulated in Table 1. To make data records more understandable, we visualize them in the form of picture of houses and it is shown in Fig. 9.

Feature name	**Possible values**	**Feature name**	**Possible values**
Length of Funnel	Short	*Texture of Door*	Black
	Long		Horizontal lined
Number of Funnel	One		Vertical lined
	Two		Checked
	Three	*Position of Door*	Left
	Four		Middle
Shape of Window	Square		Right
	Circle	*Position of plant*	None
Position of Window	Left		Left
	Middle		Right
	Right		Both
Style of Window	Cross lined	*Texture of Roof*	None
	Horizontal lined		Black
	Vertical lined		Checked

Table 1. Attributes of a house and their possible values

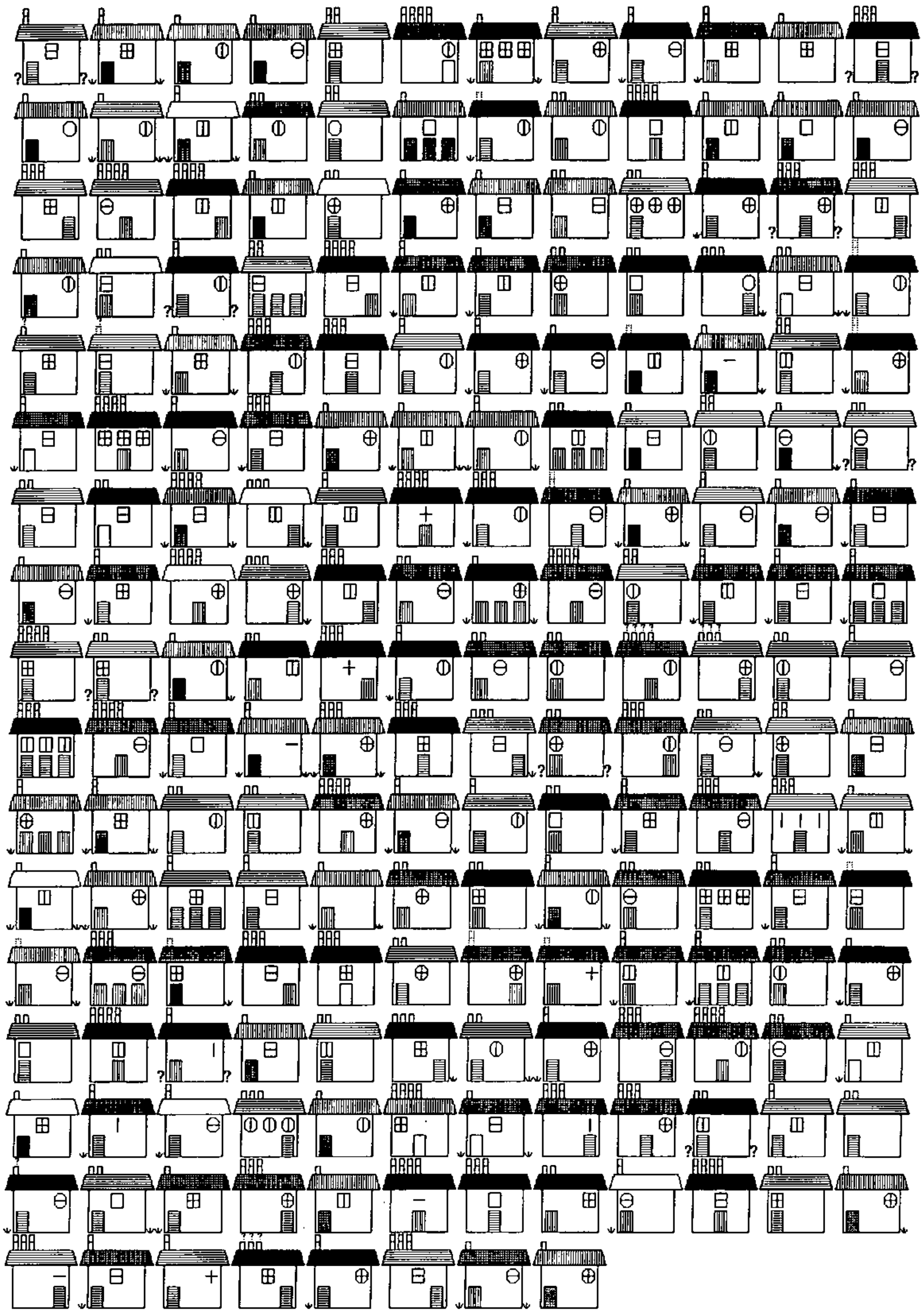


Figure 9. Simulated data of houses

During the generation of the simulated data, we had predefined all records were classified into 4 clusters according to the characteristics listed below:

Cluster 1:

Number of funnel = 4 and *Position of door* = Middle.

Number of funnel = 3 and *Position of door* = Right.

Cluster 2:

Position of plant = Left and *Length of funnel* = Long.

Position of plant = Left and *Texture of Door* = Horizontal lined.

Cluster 3:

Number of funnel = 2 and *Position of window* = Left.

Texture of roof = Horizontal and *Length of funnel* = Short.

Cluster 4:

Texture of door = Black and *Length of funnel* = Short.

Texture of door = Checked and *Length of funnel* = Short.

Position of plant = Both and *Texture of roof* = Vertical lined.

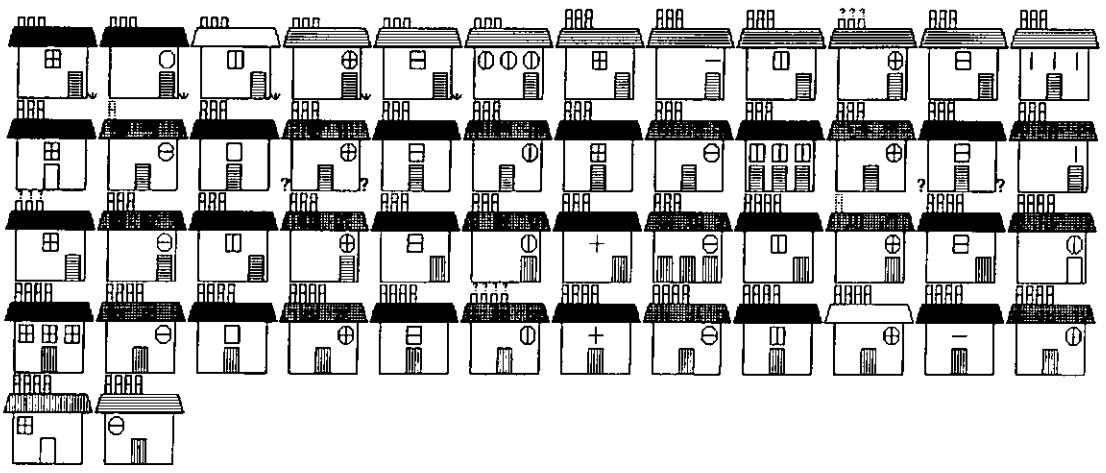
To simulate the probabilistic environment, we therefore intentionally introduced 10% to 20% noises in different attributes. Also, to ensure that the data contains missing values, 5% of the data are randomly deleted (represented by "?" or not displayed in Fig. 9). Furthermore, to illustrate the capability of identifying relevant feature values from irrelevant ones, two features are randomly generated at all. They are *Style of Window* and *Shape of Window*. Note that there are overlapping clusters generated in the simulated data, e.g., the eighth house from left to right of the second row and the eighth house from left to right of the third row. They contain the characteristics of both clusters 3 and 4.

We tested this set of simulated with different approaches and the accuracy is tabulated in Table 2.

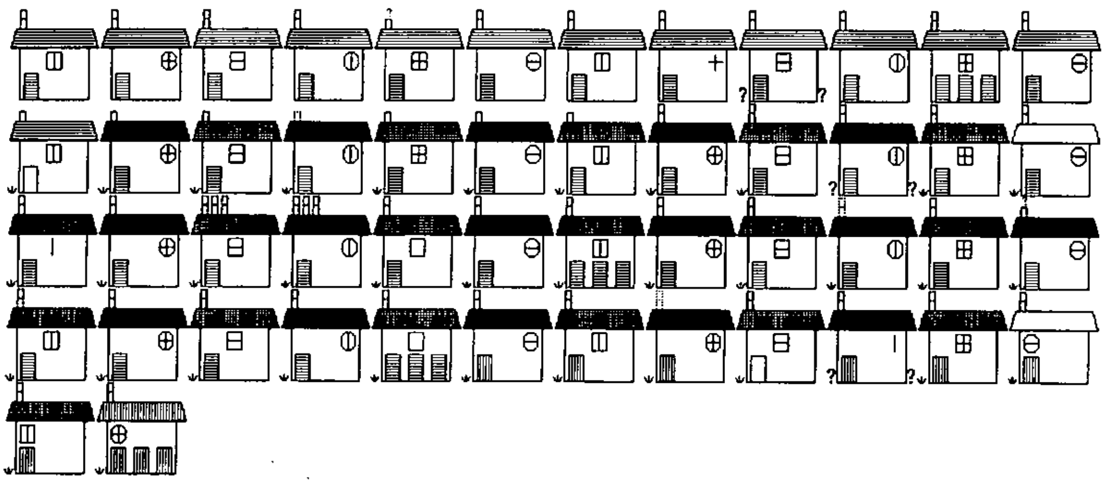
	Accuracy	Rank
Our algorithm	99.00%	1
COBWEB	53.50%	6
AUTOCLASS	97.50%	2
k-means	66.00%	4
k-modes	62.50%	5
Condorcet	92.50%	3

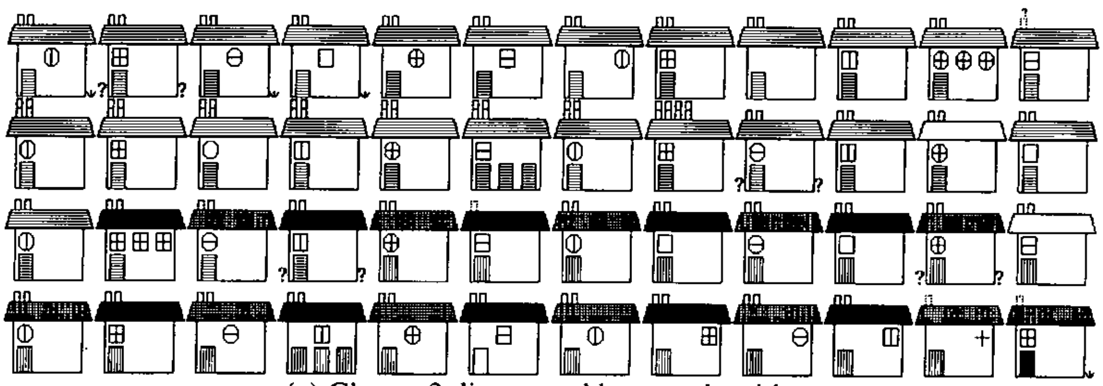
Table 2. Accuracy for the simulated data.

As shown in Table 2, our algorithm performed better than COBWEB, *k*-means, and *k*-modes. The accuracy of AUTOCLASS, Condorcet and our algorithm is higher than 90%. The performance of COBWEB, *k*-means, and *k*-modes algorithm is relatively poor than the rest algorithms because the simulated data contain both noisy and missing values. Even we have substituted the missing values by the mode value, this may also incur distortion and affects their performance. The performance of AUTOCLASS is more or less the same to our algorithm. The accuracy of our algorithm is only 1.5% greater than that of AUTOCLASS. It should be noted that the number of clusters discovered by COBWEB and Condorcet is different from that in original simulated data. COBWEB and Condorcet discovered 3 and 9 clusters respectively. The accuracy is thus obtained by applying the procedure described at the beginning of this section. To visually depict all the clustering results, the figures of houses are shown in Fig. 10(a-d) for our algorithm.



(a) Cluster 1 discovered by our algorithm


(b) Cluster 2 discovered by our algorithm


(c) Cluster 3 discovered by our algorithm

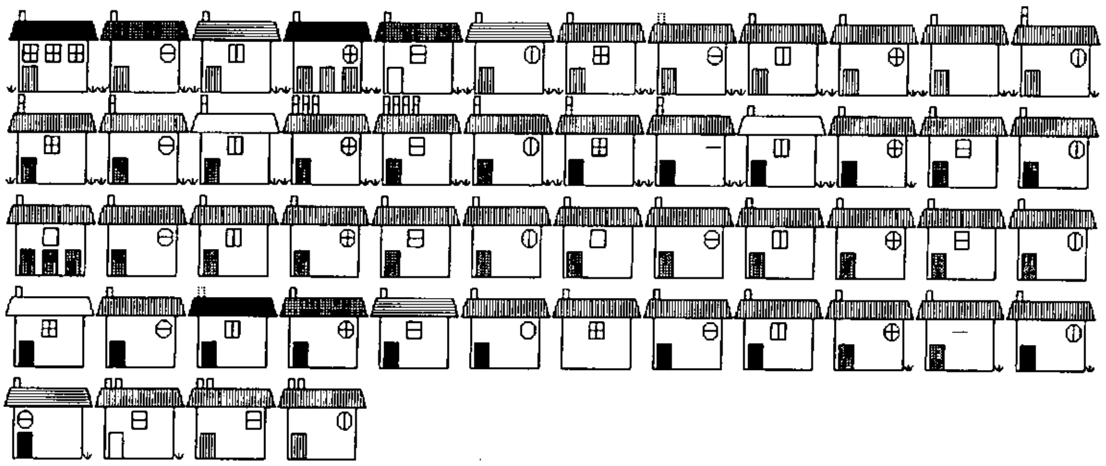

(d) Cluster 4 discovered by our algorithm

Figure 10. Results discovered by our algorithm:
(a) Cluster 1; (b) Cluster 2; (c) Cluster 3; (d) Cluster4.

In addition to the clustering result, a set of if-then rules is also constructed and some of the most interesting rules are listed in Table 3. From the table, it is observed that the rules identified are matched with the predefined patterns generated in records. It is found that Cluster 1 is mainly characterized by *Number of funnel* and *Position of door*; Cluster 2 is characterized by *Position of plant*, *Length of funnel* and *Texture of door*; Cluster 3 is characterized by *Number of funnel*, *Position of window*, *Texture of roof* and *Length of funnel*; and Cluster 4 is characterized by *Texture of door*, *Length of funnel*, *Position of plant* and *Texture of roof*.

IF *Number of funnel* = 4 and *Position of door* = Middle THEN Cluster 1 with *support* = 0.0700 and *confidence* = 1.0000
IF *Number of funnel* = 3 and *Position of door* = Right THEN Cluster 1 with *support* = 0.8500 and *confidence* = 1.0000
IF *Position of plant* = None and *Length of funnel* = long THEN Cluster 1 with *support* = 0.2000 and *confidence* = 0.4167

IF *Position of plant* = Left and *Length of funnel* = Long THEN Cluster 2 with *support* = 0.135 and *confidence* = 1.0000
IF *Position of plant* = Left and *Texture of Door* = Horizontal THEN Cluster 2 with *support* = 0.140 and *confidence* = 1.0000

IF *Number of funnel* = 2 and *Position of window* = Left THEN Cluster 3 with *support* = 0.1300 and *confidence* = 1.0000
IF *Texture of roof* = Horizontal and *Length of funnel* = Short THEN Cluster 3 with *support* = 0.0750 and *confidence* = 0.7500

IF *Texture of door* = Black and *Length of funnel* = Short THEN Cluster 4 with *support* = 0.0550 and *confidence* = 1.0000
IF *Texture of door* = Checked and *Length of funnel* = Short THEN Cluster 4 with *support* = 0.0750 and *confidence* = 1.0000
IF *Position of plant* = Both and *Texture of roof* = Vertical lined THEN Cluster 4 with *support* = 0.070 and *confidence* = 1.0000

Table 3. Some of the if-then rules generated by our clustering algorithm from the simulated data.

With the advantage of discovering overlapping clusters in our algorithm, our clustering algorithm is able to discover records that actually belong to more than one cluster. For example, the last two houses in Fig. 10d) are characterized by *texture of roof* = horizontal lined and *number of funnel* = 2. Since *texture of roof* = horizontal lined is the characteristic of cluster 4 and *number of funnel* = 2 is the characteristic of cluster 3. These two houses have positive evidence to be clustered into both cluster 3 and cluster 4.

4.2.2 Real-life Databases

This section aims at evaluating the capability of our method in solving the clustering problem in real-life databases. They are (i) the medical database; (ii) the zoo database; (iii) the DNA database; (iv) the australian database; (v) the diabetes database; (vi) the satimage database; (vii) the adult database; (viii) the PBX database; and (ix) the china database.

4.2.2.1 The medical Database

The data are concerned with the state of health of some patients in an adult-patient Intensive Care Unit (ICU) in the Forest City Hospital in Ohio [Pao *et al.* 1976 and Pao and Hu 1984]. Each patient record consists of 12 components, each of which represents a symptom derived from monitoring the central nervous system, the respiratory system, the cardiovascular system, the skin signs, and the renal system of the patient (Table 4) [Pao *et al.* 1976 and Pao and Hu 1984].

Based on these symptoms, a patient may be diagnosed, by a group of physicians, into one of four disease types according to the symptoms [Pao *et al.* 1976 and Pao and Hu 1984]. A patient may belong to (1) the chest disease group; (2) the abdominal disease group; (3) the cardiac disease group; and (4) the neurological disease group [Pao and Hu 1984]. The records of 15 patients are available for Group 1, 33 for Group 3, and 25 for both Group 3 and Group 4. Therefore, the baseline accuracy for the medical database is 33.33%. In this experiment, since *k*-means algorithm cannot handle well in discrete-valued data and the data set contains only discrete-valued attributes, there is a need to transform all the attributes to binary-valued attributes. After transformation, the number of attributes is, therefore, expanded from 12 attributes to 39 attributes. For all other approaches, the original data is directly used. The accuracy for the medical database using different approaches is tabulated in Table 5.

Central nervous system

1. Levels of consciousness:
   a) Fully alert.
   b) Will raise hand, stick out tongue, wiggle toes, etc., on command. Patient may be mentally confused but not comatose.
   c) Withdraws leg or arm, or grimmaces with painful stimulus, e.g. pressure on tail beds, hard squeezing of shoulder muscles, etc.
   d) Unresponsive.
2. Reflexes:
   a) Deep tendon reflexes intact (knee jerk, ankle jerk, biceps jerk).
   b) Deep tendon reflexes absent.
3. Pupil size and response:
   a) Normal size and reaction to light.
   b) Unequal size (if previous equal).
   c) Dilated, fixed, no light reaction.

Respiratory System

4. Rate:
   a) 10-20/mm
   b) Less than 10 or greater than 20.
5. Type:
   a) Normal chest movement.
   b) Chest does not move, but abdomen raises and falls with breathing.
   c) Gasping or irregular breathing.
6. Chest signs:
   a) Clear, coarse rhonchi, scattered rales.
   b) Widespread rales (at least one-half of one lung).
   c) Severe chest trauma, first-day thoracotomy, first-day thoracostomy tune.
   d) Pulmonary edema, untreated tension pneumothorax.

Cardiovascular system

7. Blood pressure:
   a) +20% of usual level (systolic).
   b) +-20-50% of usual level (systomic).
   c) + More than 50% of usual level (systolic or diastolic).
   d) Absent.

NB. If usual pressure is not available, use 120/80 under 50 years of age and 140/90 if 50 years of age or over. If on boundary, give the higher score.

8. Pulse:
   a) +-20%/min from usual rate.
   b) +-20-40/min from usual rate.
   c) +- More than 40/min from usual rate.
   d) Absent.

NB. If usual pulse is not available, use 80/min. If on boundary give the higher score.

9. Electrocardiogram:
   a) No significant abnormalities.
   b) All atrial arryrthmias, first- and second-degree heart block, occasional PVCs, bundle branch block, any abnormal EKG not covered elsewhere.
   c) Unifocal PVCs (more than 6/min), ischemia.
   d) Multifocal PVCs, third-degree heart block without operating pacemaker, recent M.I.
10. Venous pressure

If central venous pressure is available:
   a) 0-12 cm water.
   b) 13-25 cm water.
   c) More than 25 cm water.

If central venous pressure is not available, the following approximation is made:

With the patient lying flat, raise arm slowly until the veins of the back of the hand collapse; then, measure with a yardstick the distance from the hand to the level of the heart (approximately the anterior iliac spine).
   a) 0-18 cm.
   b) 19-35 cm.
   c) More than 35 cm.

11. Skin color, skin moisture, body temperature (rectal)

a) Pink, 37 + 1°C.
b) Pale, or jaundicated, or wet, or body temperature less than 36°C or greater than 38°C, or all four present.
c) Blue or temperature less than 33°C greater than 41°C, or both.

12. Rental system
   a) Normal kidney function and mine output.
   b) Urine output decreased but greater than 30 cc/h, or albumin or blood (except females during menstruation), or all three present.
   c) Anuria or less than 30 cc/h urine output.

Table 4. Data description of the medical database.

	Accuracy	Rank
Our algorithm	84.85%	1
COBWEB	44.44%	5
AUTOCLASS	70.71%	2
k-means	48.48%	3
k-modes	33.33%	6
Condorcet	45.45%	4

Table 5. Accuracy for the medical database.

As shown in Table 5, our algorithm outperformed all the evaluated algorithms. Although AUTOCLASS has a very high accuracy in the medical database, it actually cannot discover the correct number of clusters from this database. For AUTOCLASS, there are only three clusters discovered even there are four classes in the original classification. The accuracy is then obtained by merging the extra cluster to one of the matched clusters using the procedure described at the beginning of this section. The accuracy of COBWEB, *k*-means, and Condorcet are almost the same. There are only about 11% to 15% better than the baseline accuracy. It should be noted that *k*-modes algorithm can only discover one cluster and thus its accuracy is equal to the baseline accuracy (33.33%). The reason why only a single cluster is discovered even four cluster centers were initialized is that the distance between all records and one of the cluster modes is always the shortest. This is an expected observation of *k*-modes algorithm because many records would have the shortest distance to a particular cluster when the mode value is used to be the cluster center.

4.2.2.2 The zoo Database

The zoo database [Forsyth 1990] consists of 101 animal records and each record is characterized by 18 attributes. Since the unique name of each animal is irrelevant, it was ignored. All the 17 remaining attributes are discrete and 15 of them are boolean while the other two are multi-valued. The 15 boolean attributes are Hair, Feathers, Eggs, Milk, Airborne, Aquatic, Predator, Toothed, Backbone, Breathes, Venomous, Fins, Tails, Domestic, and Catsize. The two multi-valued attributes are Legs and Type. The domain of Legs is {0, 2, 4, 5, 6, 8} whereas the domain of Type is {mammal, bird, reptile, fish, amphibian, insect, coelenterate}.

The attribute Type has been identified as the class attribute. The baseline accuracy for the zoo database, that is, the accuracy obtained by assigning each record to the most common or the majority class in the database is 40.6%. In our experiment, since all the attributes are discrete-valued, there is a need to transform all the attributes to binary-valued attributes for *k*-means algorithm. Therefore, the number of attributes is expanded from original 16 attributes to 38 binary-valued attributes. The accuracy for the zoo database using different approaches is tabulated in Table 6.

	Accuracy	Rank
Our algorithm	97.03%	1
COBWEB	90.09%	3
AUTOCLASS	87.13%	4
k-means	47.75%	6
k-modes	61.39%	5
Condorcet	95.05%	2

Table 6. Accuracy for the zoo database.

As shown in Table 6, our algorithm performed better than all the evaluated algorithms and the accuracy of Condorcet is very close to ours. Although both COBWEB and Condorcet have very high accuracy in the zoo database, they actually cannot discover the correct number of clusters from this database. In COBWEB, there are only three clusters discovered in the first level of concept hierarchy (the first level below the root node) and there are eight clusters discovered in the second level of concept hierarchy. In this experiment, we therefore chose the second level because the difference between the number of clusters in original classification, which is seven, and the number of clusters in the second level is smaller than that in the first level. The accuracy is then obtained by merging the extra cluster to one of the matched clusters using the procedure described at the beginning of this section. Similarly, Condorcet is also unable to discover seven clusters in the zoo database. Only three clusters are obtained from it. Like COBWEB, the accuracy of Condorcet is calculated using the procedure described at the beginning of this section. The accuracy of *k*-means in this zoo database is the worst. It is because the *k*-means algorithm is designed for continuous-valued data. Since all the attributes in the zoo database are discrete-valued, *k*-means algorithm may not be good in finding the correct clusters even binarizing all attributes. In this experiment, our algorithm discovered one record to be clustered into more than one cluster.

4.2.2.3 The DNA Database

Each record in the DNA database [Noordewier, Towell, and Shavlik 1991] consists of a sequence of DNA, an instance name, and a class attribute. Since the unique name of each instance is irrelevant, it was ignored. A sequence of DNA contains 60 fields each of which can be filled by one of: A, G, T, C, D (i.e. A or G or T), N (i.e. A or G or C or T), S (i.e. C or G), and R (i.e. A or G). The class attribute concerns with the splice junctions that are points on a DNA sequence at which "superfluous" DNA is removed during the process of protein creation. It indicates the boundaries between exons (the parts of the DNA sequence retained after splicing) and itrons (the parts of the DNA sequence that are spliced out) and can be one of EI, IE, and N.

There are 3,190 records in total. The baseline accuracy for the DNA database, that is, the accuracy obtained by assigning each record to the most common or the majority class in the database is 50%. Since *k*-means algorithm cannot handle well in discrete-valued data and the data set contains only discrete-valued attributes, there is a need to transform all the attributes to binary-valued attributes. After transformation, the number of attributes is, therefore, expanded from 60 attributes to 287 attributes. For all other approaches, the original data is directly used. The accuracy for the DNA database using different approaches is tabulated in Table 7.

	Accuracy	Rank
Our algorithm	96.77%	1
COBWEB	64.29%	3
AUTOCLASS	73.23%	2
k-means	50.00%	4
k-modes	40.91%	5
Condorcet	0.00%	6

Table 7. Accuracy for the DNA database.

From the above results, our algorithm outperformed all the evaluated algorithms. It should be noted that the *k*-means algorithm could only discover one cluster even three random cluster centers were preset during initialization. Therefore, the accuracy for *k*-means is equal to the baseline accuracy (i.e. 50.00%). The accuracy of *k*-modes is below the baseline accuracy. It is because the cluster center initialization proposed in [Huang 1997b] may lead to many records belonging to a specific cluster center. For this reason, the *k*-modes algorithm may converge to a sub-optimal position. In addition, the performance of Condorcet is the worst among all approaches in the DNA database experiment. There are 2331 clusters discovered by Condorcet out of 3190 records. The reason why excessive creation of new clusters occurs in Condorcet is that many records have negative overall votes to all existing clusters. It seems that there is a difficulty in controlling the number of clusters in Condorcet when there are many negative votes calculated during the evaluation of pairs of records. Therefore, the accuracy of Condorcet cannot be obtained due to the excessive number of clusters. In this experiment, our algorithm discovered 42 records to be clustered into more than one cluster.

4.2.2.4 The australian Database

The australian database [Quinlan 1987] concerns credit card applications and the aim is to devise rules for assessing applications for credit cards. It consists of a class and 14 non-class attributes. Out of these non-class attributes, 6 are continuous while 8 are discrete. All attribute names and values have been changed by the donor of this database to meaningless symbols to protect confidentiality of the data. There are 690 records in total and 5% of them have one or more missing values. The baseline accuracy for this database is 55.5%. In our experiment, since *k*-means algorithm cannot handle well in discrete-valued data and there exists some attributes which are in discrete-valued domain, there is a need to transform those discrete-valued attributes to binary-valued attributes. After transformation, the number of attributes is, therefore, expanded from 15 attributes to 43 attributes. Those continuous-valued attributes are then discretized into 4 intervals for our algorithm using equal-depth discretization so that the number of records falling into each interval is the same. For COBWEB, *k*-modes and Condorcet, these continuous-valued attributes are also discretized into 4 equal-depth intervals because they are developed to handle discrete-valued attributes only. The accuracy for the australian database using different approaches is tabulated in Table 8.

	Accuracy	Rank
Our algorithm	79.99%	1
COBWEB	74.35%	3
AUTOCLASS	55.36%	5
k-means	55.50%	4
k-modes	78.26%	2
Condorcet	0.00%	6

Table 8. Accuracy for the australian database.

From the above results, our algorithm outperformed AUTOCLASS, *k*-means and Condorcet and was slightly better than both COBWEB and *k*-modes. The performance of COBWEB may be affected due to the presence of missing values in the australian database. Once noisy or missing values exist in data, they may lead to the overfitting in COBWEB [Gennari, Langley and Fisher 1990]. It should be noted that the *k*-means algorithm could only discover one cluster even two random cluster centers were generated during initialization. Therefore, the accuracy for *k*-means is equal to the baseline accuracy (i.e. 55.50%). The accuracy of AUTOCLASS for the australian database is below the baseline accuracy. It is because AUTOCLASS biases on the continuous-valued attributes when both discrete- and continuous-valued attributes constitute the australian database. If this is the case, the accuracy would probably be affected. In addition, the performance of Condorcet is the worst among all approaches in the australian database experiment. There are 21 clusters discovered by Condorcet out of 690 records. The reason why excessive creation of new clusters occurs in Condorcet is that many records have negative overall votes to all existing clusters. It seems that there is a difficulty in controlling the number of clusters in Condorcet when there are many negative votes calculated during the evaluation of pairs of records. Therefore, the accuracy of Condorcet cannot be obtained due to the excessive number of clusters.

4.2.2.5 The diabetes Database

The diabetes database [Smith *et al.* 1988] contains 768 patient records and each record is characterized by 9 attributes. Among these attributes, one of them is discrete whereas all the remaining attributes are continuous. The discrete-valued attribute is used as the class attribute and it can take value "1" (tested positive for diabetes) or "2" (tested negative for diabetes). The remaining attributes are continuous in nature and represent number of times pregnant, plasma glucose concentration in an oral glucose tolerance test, diastolic blood pressure (mm/Hg), triceps skip fold thickness (mm), 2-hour serum insulin (mu U/ml), body mass index (kg/m2). Diabetes pedigree function, and age (years). The baseline accuracy for the diabetes database is 65.1%. As all the attributes are in continuous-valued domain, it is not necessary to take any transformation of attributes before they are used for *k*-means algorithm. For our algorithm, those continuous-valued data are discretized into 4 intervals using equal-depth discretization so that the number of records falling into each interval is the same. These continuous-valued attributes are also discretized into 4 equal-depth intervals in the same way for COBWEB, *k*-modes, and Condorcet because they are developed to deal with discrete-valued attributes only. The accuracy for the diabetes database using different approaches is tabulated in Table 9.

	Accuracy	Rank
Our algorithm	68.10%	1
COBWEB	67.45%	2
AUTOCLASS	58.20%	4
k-means	65.10%	3
k-modes	54.43%	5
Condorcet	0.00%	6

Table 9. Accuracy for the diabetes database.

From the above results, our algorithm performed better than all the evaluated algorithms. The performance of COBWEB may be affected due to the presence of missing values in data. Once noisy and missing values exist in data, they may lead to the overfitting in COBWEB [Gennari, Langley and Fisher 1990]. It should be noted that the *k*-means algorithm, as the same case in australian database, could only discover one cluster even there are two random cluster centers were generated during initialization. Therefore, the accuracy for *k*-means is equal to the baseline accuracy (i.e. 65.10%). The accuracy of AUTOCLASS and *k*-modes for the australian database are below the baseline accuracy (i.e. 65.10%) by about 8% and 11% respectively. It is because the cluster center initialization proposed in *k*-modes [Huang 1997b] may lead to many records belonging to a specific cluster whose cluster centers contain many mode values of different attributes. Because of this, the *k*-modes algorithm may converge to a sub-optimal position. In addition, the performance of Condorcet is the worst among all approaches in the diabetes database experiment. There are 64 clusters discovered by Condorcet out of 768 records. The reason why excessive creation of new clusters occurs in Condorcet is that many records have negative overall votes to all existing clusters. It seems that there is a difficulty in controlling the number of clusters in Condorcet when there are many negative votes calculated during the evaluation of pairs of records. Therefore, the accuracy of Condorcet cannot be obtained due to the excessive number of clusters.

4.2.2.6 The satimage Database

The satimage database is extrated from the Landset satellite data generated from NASA by the Australian Center for Remote Sensing and used for research at the University of New South Wales. The satimage database which is a sub-area of a scene consist of 82 * 100 pixels and each of such pixels covers an area on the ground of approximately 80 * 80 m^2. Each record in the data corresponds to a 3 * 3 square neighborhood of pixels completely contained within the sub-area. Each record contains the pixel values in the four spectral bands of each of the 9 pixels in the 2 * 2 neighborhood and the class of the central pixel which was one of: red soil, cotton crop, grey soil, damp grey soil, soil with vegetation stubble, and very damp grey soil. All the 36 (= 4 spectral bands * 9 pixels in neighborhood) non-class attributes are continuous and in the range between 0 and 255. The satimage database contains 4,425 and 2,000 records in the training and the testing data sets respectively. Before performing the clustering task on the satimage database, we merge the training and testing data into a single set of data. Therefore, there are a total of 6425 records in the satimage database. The baseline accuracy for this database is 23.82%.

In this set of data, all the attributes are in continuous-valued domain. It is, therefore, not necessary to take any transformation of attributes before they are used for *k*-means algorithm. For our algorithm, those continuous-valued data are discretized into 4 intervals using equal-depth discretization so that the number of records falling into each interval is the same. These continuous-valued attributes are also discretized into 4 equal-depth intervals for COBWEB, *k*-modes, and Condorcet because they are developed to deal with discrete-valued attributes only. The accuracy for the satimage database using different approaches is tabulated in Table 10.

	Accuracy	Rank
Our algorithm	82.72%	1
COBWEB	75.75%	2
AUTOCLASS	70.26%	4
k-means	74.83%	3
k-modes	56.24%	5
Condorcet	0.00%	6

Table 10. Accuracy for the satimage database.

From the above results, the accuracy of COBWEB, AUTOCLASS, *k*-means and our algorithm are in a range between 70% to 80% and that of *k*-modes and Condorcet are the worst. In COBWEB, there are only three clusters discovered in the first level of concept hierarchy and eight clusters discovered in the second level of concept hierarchy. In this experiment, we, therefore, chose the second level because the difference between the number of classes in original classification (6 classes) and that in the second level of hierarchy is smaller than that in the first level. The accuracy is then obtained by merging the extra two clusters using the procedure described at the beginning of this section. The performance of *k*-means is relatively good for this set of data when compared to all the experiments that we have taken. The main reason is that all the attributes in the satimage database are in continuous-valued domain which is suitable for the distance measure defined in *k*-means. It should be noted that the accuracy of *k*-modes for the satimage database is not as good as COBWEB, AUTOCLASS, *k*-means and our algorithm. It is because the cluster center initialization proposed in [Huang 1997b] may not be able to locate the optimal cluster centers. If this is the case, the *k*-modes algorithm may converge to a sub-optimal position. In addition, the performance of Condorcet is the worst among all approaches in the satimage database experiment. There are 87 clusters discovered by Condorcet out of 6425 records. The reason why excessive creation of new clusters occurs in Condorcet is that many records have negative overall votes to all existing clusters. It seems that there is a difficulty in controlling the number of clusters in Condorcet when there are many negative votes calculated during the evaluation of pairs of records. Therefore, the accuracy of Condorcet cannot be obtained due to the excessive number of clusters. In this experiment, our algorithm discovered 238 records to be clustered into more than one cluster.

4.2.2.7 The adult Database

The adult database [Kohavi 1996] originally belongs to the US Census bureau. This database consists of 32,561 records in the data set. Each record is characterized by 15 attributes. Of these attributes, 6 of them are continuous. They include Age, Fnlwgt, Education-num, Capital-gain, Capital-loss, and Hours-per-week. The 9 remaining attributes are all discrete and they are given in Table 11.

Attribute	Domain
Workclass	{Private, Self-emp-not-inc, Self-emp-inc, Federal-gov, Local-gov, State-gov, Without-pay, Never-worked}
Education	{Bachelors, Some-college, 11th, HS-grad, Prof-school, Assoc-acdm, Assoc-voc, 9th, 7th-8th, 12th, Masters, 1st-4th, 10th, Doctorate, 5th-6th, Preschool}
Marital-status	{Married-civ-spouse, Divorced, Never-married, Separated, Widowed, Married-spouse-absent, Married-AF-spouse}
Occupation	{Tech-support, Craft-repair, Other-service, Sales, Exec-managerial, Prof-specialty, Handlers-cleaners, Machine-op-inspct, Adm-clerical, Farming-fishing, Transport-moving, Priv-house-serv, Protective-serv, Armed-Forces}
Relationship	{Wife, Own-child, Husband, Not-in-family, Other-relative, Unmarried}
Race	{White, Asian-Pac-Islander, Amer-Indian-Eskimo, Other, Black}
Sex	{Female, Male}
Native-country	{United-States, Cambodia, England, Puerto-Rico, Canada, Germany, Outlying-US(Guam-USVI-etc), India, Japan, Greece, South, China, Cuba, Iran, Honduras, Philippines, Italy, Poland, Jamaica, Vietnam, Mexico, Portugal, Ireland, France, Dominican-Republic, Laos, Ecuador, Taiwan, Haiti, Columbia, Hungary, Guatemala, Nicaragua, Scotland, Thailand, Yugoslavia, El-Salvador, Trinadad&Tobago, Peru, Hong, Holand-Netherlands}
Salary	{>50K, ≤50K}

Table 11. The discrete attributes in the adult database.

The attribute Salary is used as the class attribute and the baseline accuracy for the adult database is 76.1%. In our experiment, since *k*-means algorithm cannot handle well in discrete-valued data and there exists some attributes which are in discrete-valued domain, there is a need to transform all the attributes to binary-valued attributes. After transformation, the number of attributes is, therefore, expanded from 15 attributes to 62 attributes. Those continuous-valued attributes are then discretized into 4 intervals for our algorithm using equal-depth discretization so that the number of records falling into each interval is the same. For COBWEB, *k*-modes and Condorcet, the continuous-valued attributes are also discretized into 4 equal-depth intervals because they are developed to handle discrete-valued attributes only. In addition to these transformations of data, there is a need to handle missing values in this database. It is because *k*-means, *k*-modes, and Condorcet are unable to handle missing values with their distance functions. For any missing values in continuous-valued attribute, we substitute the mean value of the corresponding attribute to each missing value. As with continuous-valued data, all missing values in discrete-valued attributes are substituted by the mode value of the corresponding attribute to each missing value. The accuracy for the adult database using different approaches is tabulated in Table 12.

	Accuracy	Rank
Our algorithm	79.29%	1
COBWEB	71.60%	4
AUTOCLASS	77.97%	2
k-means	51.59%	5
k-modes	74.99%	3
Condorcet	0.00%	6

Table 12. Accuracy for the adult database.

From the above results, only AUTOCLASS and our algorithm are above the baseline accuracy (76.1%). Since the adult database contains many noisy and missing values, the clustering environment becomes extremely probabilistic. Under this environment, the accuracy of COBWEB, *k*-means and *k*-modes are all below the baseline accuracy (76.1%). The performance of COBWEB may be affected due to the presence of noisy and missing values in the database. Once noisy data exist, they may lead to the overfitting in COBWEB [Gennari, Langley and Fisher 1990]. Except Condorcet, which performed the worst in this database, *k*-means has a relatively low accuracy when compared with COBWEB, *k*-modes, and AUTOCLASS. It is because the adult database contains both continuous- and discrete-valued. Even binarizing all discrete-valued attributes to binary-valued attributes can avoid the incapability of handling discrete-valued data in *k*-means algorithm, the performance of k-means is still not very good. There is only a small difference between the accuracy of k-modes and the baseline accuracy. From the result of k-modes, it is observed that many records were assigned to a particular cluster. The most probable reason is that the cluster center defined by mode values in k-modes may often lead to many records having shortest distance to such cluster center. In addition, there are 129 clusters discovered by Condorcet out of 32561 records. The reason why excessive creation of new clusters occurs in Condorcet is that many records have negative overall votes to all existing clusters. It seems that there is a difficulty in controlling the number of clusters in Condorcet when there are many negative votes calculated during the evaluation of pairs of records. Therefore, the accuracy of Condorcet cannot be obtained due to the excessive number of clusters.

4.2.2.8 The PBX Database

The PBX database is collected from a private branch exchange (PBX) system used in a telecommunication company. A PBX is a multiple-line business telephone system that resides on the company premises. One of the significant features of the PBX is its ability to record call activity, for example, it can make records of all calls and callers. The PBX database is composed of a set of phone call records concerning the usage of the PBX in the telecommunication company. There are 3,009 records in this database. Each record is characterized by 13 attributes: *Time-of-call-origination*, *Duration-of-call*, *Calling-party-identification*, *Originating-extension-number*, *Trunk-identification*, *Trunk-number*, *Trunk-access-code*, *Directory-number-dialed*, *Account-code*, *Tonant-number*, *Metering-group*, *Call-charge*, and *Modem-identification-number*. Except for two attributes *Calling-party-identification* and *Trunk-identification* that are discrete, all the remaining attributes are continuous-valued attributes. The domain of *Calling-party-identification* is {ST, AT, TI, DD, DS, DT, CO, LN, TL} and the domain of *Trunk-identification* is {-, C, F, L, W, T}. There are plenty of missing values in this database, in particular, 98.4% of records have missing values in at least one of the attributes.

In the PBX database, the attribute Calling-party-identification is chosen as the class attribute and the baseline accuracy for this database is 65.9%. In our experiment, since *k*-means algorithm cannot handle well in discrete-valued data and there exists some attributes which are in discrete-valued domain, there is a need to transform all the attributes to binary-valued attributes. After transformation, the number of attributes is, therefore, expanded from 9 attributes to 12 attributes. Those continuous-valued attributes are discretized into 4 intervals for our algorithm using equal-depth discretization so that the number of records falling into each interval is the same. For COBWEB, *k*-modes and Condorcet, these continuous-valued attributes are also discretized into 4 equal-depth intervals because they are developed to handle discrete-valued attributes only. In addition to these transformations of data, there is a need to handle missing values in this database. It is because *k*-means, *k*-modes, and Condorcet are unable to handle missing values with their distance functions. For any missing values in continuous-valued attribute, we substitute the mean value of the corresponding attribute to each missing value. As with continuous-valued data, all missing values in discrete-valued attributes are substituted by the mode value of the corresponding attribute to each missing value. The accuracy for the PBX database using different approaches is tabulated in Table 13.

	Accuracy	Rank
Our algorithm	97.55%	1
COBWEB	55.65%	5
AUTOCLASS	60.60%	4
k-means	53.05%	6
k-modes	78.40%	2
Condorcet	65.90%	3

Table 13. Accuracy for the PBX database.

From the above results, our algorithm outperformed all of the evaluated algorithms. In particular, COBWEB, AUTOCLASS and *k*-means are all below the baseline accuracy. The PBX database contains many noisy and missing values and thus the clustering environment becomes very probabilistic. As a consequence, all these three algorithms perform not very effective in this probabilistic and noisy environment. As for COBWEB, once noisy data exist, they may lead to the overfitting [Gennari, Langley and Fisher 1990]. In addition, COBWEB can only discover three clusters in the first level of the concept hierarchy whereas there are only two clusters in the original database. The accuracy of COBWEB is thus obtained by merging the extra cluster to one of the rest clusters using the procedure described at the beginning of this section. For the performance of *k*-means algorithm, the result shows that it is unable to perform well even with binarized discrete-valued attributes. In addition, there is only one cluster discovered by Condorcet. The reason for this observation is that the overall vote is always positive for all records to the existing cluster. All records are therefore merged to that cluster. Therefore, the accuracy of Condorcet is equal to the baseline accuracy (65.90%).

4.2.2.9 The china Database

The china database is provided by the China Business Center of The Hong Kong Polytechnic University. It contains the data collected in a survey performed by the census bureau of the government in People's Republic of China. This survey concerns with the situations of production and operation of 800 industrial enterprises in Mainland China in 1992. The china database, in turn, contains 800 records representing the details of these industrial enterprises. Each record is characterized by 56 attributes. Some of the records in the database contain some missing values. The percentages of records having missing values in each attribute range from 0.5% to 97.5%. These attributes, percentages of records having missing values in them, and their descriptions are given in Table 14.

One of these attributes, called A1, is identified as the class attribute. For those records containing missing value in attribute A1, they are deleted before experiment. Attribute A1 is then discretized into 4 intervals using equal-depth discretization. Consequently, there are one-forth of records in each class. In other words, the baseline accuracy for the china database is therefore 25%. In our experiment, since *k*-means algorithm cannot handle well in discrete-valued data and there exists some attributes which are in discrete-valued domain, there is a need to transform all the attributes to binary-valued attributes. After transformation, the number of attributes is, therefore, expanded from 55 attributes to 67 attributes. Those continuous-valued attributes are then discretized into 4 intervals for our algorithm using equal-depth discretization so that the number of records falling into each interval is the same. For COBWEB, *k*-modes and Condorcet, these continuous-valued attributes are also discretized into 4 equal-depth intervals because they are developed to handle discrete-valued attributes only. In addition to these transformations of data, there is a need to handle missing values in this database. It is because *k*-means, *k*-modes, and Condorcet are unable to handle missing value for their distance function. For any missing values in continuous-valued attribute, we substitute the mean value of the corresponding attribute to each missing value. As with continuous-valued data, all missing values in discrete-valued attributes are substituted by the mode value of the corresponding attribute to each missing value. The accuracy for the china database using different approaches is tabulated in Table 15

Attribute	% missing	Description
K4	0.5	Type of enterprise
K5	1.3	Code of industry
K8	0.5	Size of enterprise
K9	12.4	Form of business affiliated
A1	3.8	Total value of industrial output (constant price of 1990)
A2	31.1	Total value of Government-planned product
A3	3.8	Total value of industrial output (current year's price)
A4	43.9	Total value of exports output
A5	4.3	Industrial net output
A6	5.5	Total production expenses
A7	5.5	Materials purchased (indenter's materials included)
A8	12.1	Fuel purchased
A9	91	Special tax on oil burning
A10	7.1	Power purchased
A11	5.6	Wage
A12	6.0	Withdrawal of employee's welfare fund
A13	5.8	Depreciation cost
A14	10.1	Withdrawal of allocations of major overhauls
A15	7.1	Interest exchange
A16	13.6	Real money
A17	7.8	Other cost: physical ware and tear
A18	7.6	Other cost: non-physical ware and tear
A19	4.4	Sales revenue
A20	46.0	Exports sales revenue
A21	7.3	Sales dollar
A22	4.1	Manufacturing cost of good sold
A23	7.0	Sales and other cost
A24	38.5	Sales cost: physical wear and tear
A25	19.5	Sales cost: non-physical wear and tear
A26	92.3	Cost of technology transfer
A27	56.3	Cost of technology improvement
A28	5.4	Sales profit
A29	24.9	Non-operating earning
A30	4.8	Non-operating cost
A31	97.5	Resources tax
A32	7.4	Gross profit
A33	90.8	Single retained profit
A34	69.9	Loan payable before tax
A35	37.5	Interest tax payable
A36	50.9	Income tax

A37	45.0	Business retained profit
A38	4.3	Gross payroll
A39	10.8	Incentive payment
A40	4.4	Average number of employees
A41	6.1	Normal working hours
A42	28.1	Idle time
A43	89.8	Original value of fixed assets at term-end
A44	94.4	Fixed assets to net worth at term-end
A45	94.3	Total liquid assets
A46	92.0	Determinable sub-total of liquid assets
A47	92.5	Savings capital
A48	95.0	Production capital
A49	95.0	Capital of end products
A50	94.9	Other sub-total of liquid assets
A51	96.4	Goods shipped in transit
A52	95.4	Loan receivable and cash in advance

Table 14. Attributes in the china database.

	Accuracy	Rank
Our algorithm	90.39%	1
COBWEB	80.26%	2
AUTOCLASS	71.30%	3
k-means	29.35%	5
k-modes	27.66%	6
Condorcet	48.83%	4

Table 15. Accuracy for the china database.

From the above results, our algorithm performed better than all evaluated algorithms. The accuracy of our algorithm is about 10% and 20% better than that of COBWEB and AUTOCLASS respectively. The china database contains many noisy and missing values, the clustering environment becomes extremely probabilistic. Therefore, the performance of both COBWEB and AUTOCLASS algorithms may be affected in such probabilistic and noisy environment. As for COBWEB, once noisy and missing values exist in data, this may lead to the overfitting [Gennari, Langley and Fisher 1990]. In addition, COBWEB discovered two and six clusters in the first and second level of the concept hierarchy respectively and there are only four clusters in the original database. In this case, we chose the result with six clusters for evaluation because the accuracy is higher for six clusters than that of two clusters. The accuracy of COBWEB is obtained by merging the extra two clusters to the matched clusters using the procedure described at the beginning of this section. It should be noted that the accuracy of both *k*-means and *k*-modes algorithms for the china database are only a little better than the baseline accuracy (25.00%). In other words, the results for these two algorithms are almost in a random manner. It is because the cluster centers that they converged may not be the optimal location. One of the reasons why *k*-means algorithm cannot perform well is that the database contains both continuous- and discrete-valued data. In addition, there are six clusters discovered by Condorcet. As with COBWEB, the accuracy of COBWEB is thus obtained by merging the extra two clusters to the rest clusters using the procedure described at the beginning of this section. In this experiment, our algorithm discovered 65 records to be clustered into more than one cluster.

4.3 Discussions

Based on the experiments, we evaluated the performance of our method proposed in Chapter 3 by using several real-life databases. For comparison, we applied (i) *k*-means algorithm, a traditional clustering approaches; (ii) COBWEB, an incremental machine learning approach for clustering; (iii) AUTOCLASS, a parametric method based on Bayesian classification; (iv) *k*-modes algorithm, a clustering categorical data approach in data mining; and (v) CONDORCET, a demographic clustering approach. The experimental results show that our technique is promising. The experimental results are summarized in Table16.

	Accuracy					
	Our Algorithm	COBWEB	AUTOCLASS	*k*-means	*k*-modes	Condorcet
Medical	84.85% (1st)	44.44% (5th)	70.71% (2nd)	48.48% (3rd)	33.33% (6th)	45.45% (4th)
Zoo	97.03% (1st)	90.09% (3rd)	87.13% (4th)	47.75% (6th)	61.39% (5th)	95.05% (2nd)
DNA	96.77% (1st)	64.29% (3rd)	73.23% (2nd)	50.00% (4th)	40.91% (5th)	0.00% (6th)
Australian	79.99% (1st)	74.35% (3rd)	55.36% (5th)	55.50% (4th)	78.26% (2nd)	0.00% (6th)
Diabetes	68.10% (1st)	67.45% (2nd)	58.20% (4th)	65.10% (3rd)	54.43% (5th)	0.00% (6th)
Satimage	82.72% (1st)	75.75% (2nd)	70.26% (4th)	74.83% (3rd)	56.24% (5th)	0.00% (6th)
Adult	79.29% (1st)	71.60% (4th)	77.97% (2nd)	51.59% (5th)	74.99% (3rd)	0.00% (6th)
PBX	97.55% (1st)	55.65% (5th)	60.60% (4th)	53.05% (6th)	78.40% (2nd)	65.90% (3rd)
China	90.39% (1st)	80.26% (2nd)	71.30% (3rd)	29.35% (5th)	27.66% (6th)	48.84% (4th)
Average	86.30% (1st)	69.32% (3rd)	69.42% (2nd)	52.85% (5th)	56.18% (4th)	28.36% (6th)
S.D	0.1002	0.1350	0.1005	0.1249	0.1915	0.3638

Table 16. Summary of the experimental results.

From Table 16, we can conclude that our clustering algorithm is consistently better than all the evaluated algorithms through the results of average accuracy. The performance in terms of accuracy and rank of COBWEB and AUTOCLASS are more or less the same. Their average rank is between the second and the third. It should be noted that the standard deviation of AUTOCLASS and our algorithm is almost the same. The results of AUTOCLASS show that AUTOCLASS is consistent on their clustering result even our algorithm performs better in terms of accuracy. The overall performance of *k*-means and *k*-modes are similar. It is not difficult to explain such results because the concept of *k*-modes algorithm is very similar to that of *k*-means algorithm. With the reason that Condorcet often produced either an excessive number of clusters or a single cluster, it is quite difficult to evaluate its accuracy. The average accuracy of Condorcet is thus the lowest and the standard deviation is the highest among all evaluated algorithms.

For *k*-means algorithm in handling discrete-valued data, Hamming distance may be one of the possible solutions. There are some weaknesses if Hamming distance is employed to be the distance function for *k*-means algorithm. For example, given a set of data defined by two attributes, the possible values of distance that can be calculated using Hamming distance between a record and a cluster center is 0, 1, or 2. There would be a large number of records that has equal distance with respect to the cluster center in large data set. This would affect the performance of *k*-means algorithm because of the incapability in determining the similarity between records. Some researchers modified the *k*-means algorithm to handle discrete-valued data through binarizing all discrete-valued data to binary-valued data. Unfortunately, this approach may result in generating a large number of attributes for each discrete-valued attribute. Besides the issue in handling discrete-valued data, algorithms using a distance measure would also lead to the incapability in handling noisy and missing values. Even substituting the mean or mode values to the missing values seems solving the problem, this may incur some distortions to the original data [Matthews and Hearne 1991]. In some cases, substitution is impossible and invalid. For instance, given a clinical database, it is clearly that some attributes may only exist in male patients but not in female patients. Substituting any kind of values to the missing values is, however, theoretically incorrect with respect to the problem domain. In addition to these problems, cluster discovered by *k*-means is always convex in shape [Anderberg 1973, Jain and Dubes 1988, Everitt 1993]. This further makes *k*-means algorithm impossible to handle discrete-valued data, as there are no ordering and direction in discrete variables. Furthermore, all the traditional distance-based clustering algorithms [Anderberg 1973, Jain and Dubes 1988, Everitt 1993, and MacQueen 1967] cannot generate explanation for their clustering result. For data mining task to be effective, data interpretation is one of the vital processes in KDD [Fayyad, Piatetsky-Shapiro, and Smyth 1996a, 1996b, 1996c]. This may be difficult for users to understand what knowledge has been mined and how useful it is.

As with *k*-means, *k*-modes algorithm shares almost the same features and characteristics when compared with *k*-means. Although *k*-modes algorithm is able to handle discrete-valued data, it still cannot handle noisy or missing values inside data. In the case when Hamming distance is employed as the distance measure, there would be many records which have equal distance from their corresponding cluster center. The measure on similarity is, therefore, difficult to determine.

Similar to the traditional distance-based clustering approaches [Anderberg 1973; Jain and Dubes 1988; and Everitt 1993], Condorcet makes use of a modified hamming distance as its clustering criterion. It is, therefore, also unable to handle noisy and missing values. Although the original algorithm only aimed at handling discrete-valued data, it also claimed to be able to handle continuous-valued data as well [Cabena 1998]. This is done by setting a threshold to determine whether or not an attribute value should be assigned to either positive or negative vote. The value of such threshold, however, may not be easily determined. If the threshold is set too high, the number of positive votes that records received would be greater than that of negative votes and many records would be merged to a single cluster. If the threshold is set too low, the number of negative votes that records received would be greater than that of positive votes and many records would be split even they are very similar. Assigning an optimal threshold may be possible when there is enough domain and expert knowledge. However, unfortunately, such knowledge does not always exist in data mining task. This may result in a difficulty especially for causal users.

As stated in [Gennari, Langley and Fisher 1990], COBWEB retains all instances ever encountered as terminal nodes in its concept hierarchy. Although this approach works well in noise-free, symbolic domains, it can lead to "overfitting the data" in noisy or numeric domains. With this weakness, COBWEB may not perform its task effectively when data contain missing value or when data contain noise as overfitting problem is one of the major challenges in data mining community [Fayyad, Piatetsky-Shapiro, and Smyth 1996a, Fayyad, Piatetsky-Shapiro, and Smyth 1996b, and Fayyad 1997]. In addition to this, there is a limitation on incremental learning in COBWEB. The final concept hierarchy in COBWEB may be totally different when different orderings of input data are given. In other words, the structure and the number of branches in hierarchy would be different for the same set of data with different ordering. If highly similar records are initially encountered, several relatively large clusters will evolve, and there will be a natural tendency to replace later records, even those records that are relatively unique, into these established categories. For this reason, COBWEB uses merging and splitting operators that mitigate data ordering, but these operators cannot completely eliminate the effect [Fisher, *et.al.* 1993]. In addition, given a concept hierarchy generated, it is quite difficult to determine the optimal number of clusters. It is because there are many cut-off points in the concept hierarchy like that in traditional hierarchical clustering. Unlike hierarchical clustering techniques, which use the largest distance between two connected layers in the hierarchy as the cut-off point, the concept hierarchy in COBWEB does not have such distance measure. As a consequence, together with the ordering effect, the determination of the number of clusters becomes more difficult. Furthermore, with only a concept hierarchy as clustering result, it is difficult to draw data interpretations from it.

AUTOCLASS performs its task by using a fundamental finite mixture model to find an interclass mixture probability model that gives the chance of an instance belonging to different classes. Depending on the type of variables involved, AUTOCLASS makes different assumptions about their distributions [Cheesman, *et.al.* 1988; and Cheesman and Stutz 1995]. For example, nominal variables are assumed to have a Bernoulli distribution, real variables are assumed to have a Gaussian distribution, number counts are assumed to have a Poisson distributions, etc. With these assumptions, groupings of records are derived to maximize the posterior probability of individual clusters given the feature distribution. This maximization process allows the most probable number of classes to emerge from the data. The most probable number of classes also emerges from the data – smaller classes must exhibit greater skew in their attribute-value distributions to be accepted as clusters, thus biasing AUTOCLASS against numerous classes unless the data warrant them [Fisher, M.J. Pazzani and P. Langley 1991]. Conversely, there is a natural bias against overly large classes because value distributions will have a greater a priori tendency to match the value distributions of the population as a whole. In addition, AUTOCLASS conducts a large number of trials. The program runs on and on trying to find a better classification until it is manually stopped by user. The best results seen so far will be returned as the most probable partition structure of the data. This exhaustive kind of search makes it computationally expensive, it may take considerably greater amount of time to generate reasonable results.

4.4 Remarks

In evaluating the performance of a method, it is often necessary to consider the amount of computation effort that an algorithm requires. One measure of this effort is the complexity of the method [Clark and Niblett 1987]. Let M denotes the size of the data set and n denotes the total number of attributes describing each of the record. The time complexity of our proposed algorithm in evaluating a chromosome and identifying the interesting patterns is $O(nM)$. The overall time taken is thus $O(nM \times$ Number of chromosomes in population $\times$ Total number of iterations to converge$)$. Our experience is that the number of chromosomes is about 50 for about ten thousand records and the average total number of iterations to converge is about 100.

The critical component of the COBWEB algorithm is the process of assimilating records to the existing hierarchy. Since COBWEB employs an incremental method to build the hierarchy, the time taken to construct the complete tree depends very much on the structure of the tree. On average, it takes time $O(nM \log M)$.

As for AUTOCLASS, the time taken for a particular grouping in a single trial is only $O(nM)$. However, the number of trials to achieve a specific level of performance in AUTOCLASS is not defined very clear. The stopping criteria depend on how the user presets the parameters. The overall time complexity is therefore $O(nM \times \text{Total number of trials})$.

For traditional hierarchical clustering approaches, they are required to construct the similarity matrix to determine the next merging or splitting of records. Native implementations of agglomerative hierarchical clustering have computational complexity $O(M^3)$, but $O(M^2)$ implementations have been developed [Murtagh 1985]. As another traditional clustering approach, *k*-means and its variants perform their task much more efficient than hierarchical clustering approaches. The overall time complexity of *k*-means is $O(nkM)$ where k is the number of clusters.

For the time complexity issue, our algorithm undoubtedly outperforms the traditional hierarchical clustering approaches. Assume that the number of chromosomes, the number of iterations or trials, and the number of clusters (k) are very small compared to the number of records (M), the time complexity of *k*-means algorithm, AUTOCLASS and our clustering algorithm is $O(nM)$. However, taking into the consideration of the output result within a period of time, an explicit convergence criterion of GA in our algorithm should be more robust and reliable than the endless searching procedure in AUTOCLASS. Although the incremental learning method allows COBWEB to perform its task on a single pass, there are mathematical manipulations to calculate the score for node merging, node splitting, and node creation so that a record can be added to existing hierarchy. In addition, the incremental learning often results in a sub-optimal solution due to different orderings of records. To partially avoid the ordering effect, COBWEB requires randomizing the sequence of records before it performs its task. It should be noted that, unlike all of the existing clustering algorithms, GA employed in our algorithm can be easily implemented on parallel computer architecture and can be made computationally very efficient [Wu *et al.* 1998].

To summarize the time complexity issue, table 17 tabulates the time complexity for all the discussed approaches.

Algorithm	Time Complexity	
Our Algorithm	$O(nM$ * no. of chromosomes * no. of iterations$)$	
Hierarchical Clustering Algorithms	$O(nM^3)$ / $O(nM^2)$	
k-means	$O(nkM$ * no. of iterations$)$	
k-modes	$O(nkM$ * no. of iterations$)$	
Condorcet	1st iteration: $O(n * M * (M + 1) / 2)$	2nd or above iterations: $O(nM^2)$
COBWEB	$O(nM * \log M)$	
AutoClass	$O(nM$ * no. of iterations$)$	

Table 17 Time Complexity of different algorithms.

In above table, k denotes the number of clusters. In addition, n and M denote the number of attributes and number of records in database respectively. Except for the hierarchical clustering algorithms and Condorcet (second or above iterations), which taking M^2 or even M^3 in time complexity with respect to the number of records, all other listed algorithms are linear in time complexity with respect to the number of records.

In addition to the theoretical discussion on the time complexity for our proposed clustering algorithm, an experiment on the time complexity was also taken to provide the proof. Ten sets of data, each of them characterized by ten attributes, were generated for the experiment. The number of records for the first set of data is 10,000. For each time that we generated the next set of data, 10,000 records are added to the previous set. Therefore, the number of records in the last set of data is 100,000.

We then applied our proposed clustering algorithm on the ten sets of generated data using a personal computer with CPU power of Pentium 233 and 32MB main memory. During the test, no other processes were running on the personal computer. Table 18 and figure 11 show the time taken for each set of data.

From figure 11, we can observe that the time taken is approximately linear to the number of records. It should be noted that there is a slightly non-linear between number of records equal 40,000 and 50,000. The most likely reason is that the main memory is not enough for the total memory that needed to store all the records, chromosomes and all temporary memories for computations. This may result in some overheads with the use of virtual memory. But, generally, the graph is able to illustrate that our proposed algorithm is linear in time complexity with respect to number of records.

Number of records	Time taken (in hours)
10000	2.96
20000	5.32
30000	7.77
40000	10.42
50000	15.29
60000	19.53
70000	22.89
80000	25.09
90000	28.55
100000	31.56

Table 18. The time taken using the proposed algorithm against number of records in database.

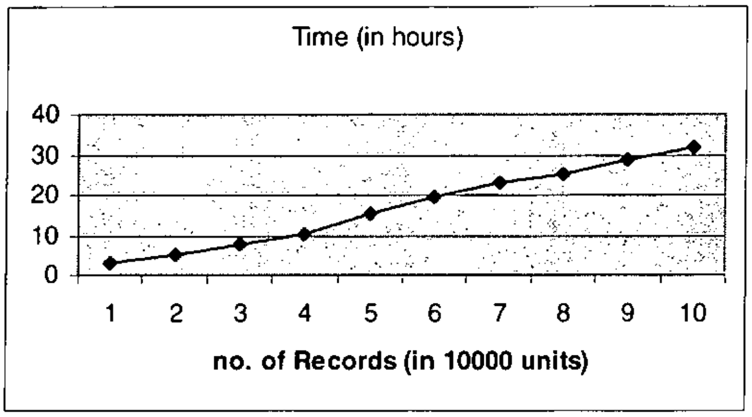


Figure 11. Graph showing the time taken using the proposed algorithm against number of records in database.

For algorithms that we have used for performance evaluation, some of them cannot easily be figured out how long that they have taken for a given data set. For example, Condorcet generated 2331 clusters out of 3190 records after 4 iterations in the DNA database in section 4.2.2.3. The time taken for these 4 iterations is less than half an hour. It is obvious that the solution would be totally corrupted no matter how long Condorcet continues the search process. Because of this reason, the time is quite, to a certain extent, meaningless to evaluate accurately.

As for COBWEB, the performance in terms of time complexity is the fastest among all approaches that we evaluated. Given a database, the time taken for each trial using the same database is almost the same. However, the accuracy for different trials with different ordering of records deviates a lot. For instance, the worst accuracy for the DNA database is only 42.60%. Comparing with the highest accuracy (64.29%) that is reported in section 4.2.2.3, there is more than 20% accuracy difference.

As described in the beginning of this section, the search process of AUTOCLASS is endless. It is sometimes unable to determine when to stop the search process. Intuitively, the longer the time taken for the search process of an algorithm, the better the performance should be obtained. Once again, the DNA database described in section 4.2.2.3 took more than 10 hours to complete 500 trials in a Sun Ultra 5 machine with 256MB main memory with average accuracy of 73.23% whereas it only took around 6 hours for our proposed algorithm with accuracy of 96.77% using the same machine. It is unknown how much accuracy would be improved if longer time is given to AUTOCLASS. Because of this, it is difficult to measure the actual time taken for the search process in AUTOCLASS.

Although incorporating genetic algorithm in our proposed algorithm would be easy and robust in searching the optimal solution, there is one weakness. If there is case that the number of records in database is very large and the time is critical to the application, our proposed algorithm may not be very suitable. It is because genetic algorithm requires a numerous number of iterations so as to perform its task.

To overcome the problem in genetic algorithm, or in other words to shorten the time needed so as to increase the performance of our proposed algorithm, we performed an experiment to determine the impact of data sampling on our algorithm. Data sampling has been proposed for mining association rules [Zaki *et al.* 1997 and Lin and Dunham 1998] to speed up the mining process by reducing I/O costs and the number of records to be considered. In the experiment, a database with 100,000 records was generated and 5 different sampling percentage values were evaluated. The evaluated percentages are 100% (no sampling), 50%, 10%, 5% and 1%. For each sampling percentage, 20 sets of sampled records were extracted in a random manner. Figure 12 depicts the average accuracy of different sampling percentage.

As shown in figure 12, there is a significant drop in accuracy when the sampling percentage is down from 5% to 1%. The average accuracy is more or less the same when sampling accuracy is between 100% and 5%. This indicates that the accuracy using only 5% of sampled data is almost as good as that of using whole data set. Unlike most of the clustering algorithms [Jain and Dubes 1988; Everitt 1993, Huang 1997b, Michaud 1997, Fisher 1987, Cheesman *et al.* 1988, and Cheesman and Stutz 1995], which have to use the whole database in order to assign each record to a cluster, our algorithm can easily assign each record in database to a cluster using only the clustering result of 5% of sampled data. This is done by the classification method [Chan, Wong and Chiu 1988 and Chan and Wong 1990]. With the fact that our proposed clustering algorithm is linear in time complexity with respect to the number of records, the efficiency of the proposed algorithm can enhance significantly using random sampling technique.

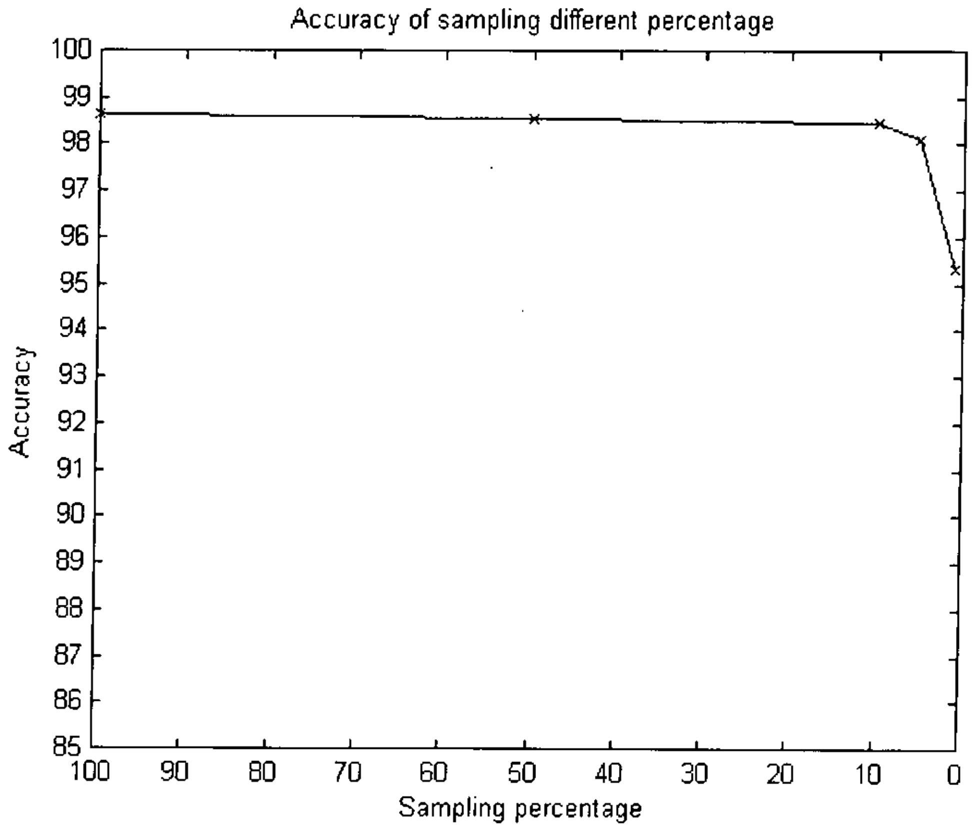


Figure 12. The average accuracy of different sampling accuracy

In addition to the computation complexity issue, it should be noted that one of the characteristics of Condorcet is the ability to determine the number of clusters from the given data automatically. Because of this, user cannot force the algorithm to discover a specific number of clusters. Our clustering algorithm allows user to pre-set the number of clusters as input and discovers the most meaningful grouping according to the requirement of user. In other words, our clustering is similar to *k*-means and *k*-modes in this case.

Chapter 5

Data Transformation

Given the database systems consisting of transaction data (such as records of purchase, electronic fund transfer or phone calls, etc.) or relational data (such as customer information, inventory records, and credit histories, etc.) or both of them, our goal is to perform clustering and discover interesting patterns underlying them. Since it is not unusual for both transaction and relational data to be collected in many database systems such as point-of-sale system, electronic fund transfer system, and telecommunication system, etc., it is important that this problem should be dealt with effectively. To do so, we define a set of transformation functions for different types of attributes in databases containing both transaction and relational data. Based on these functions, we can cluster the given set of data into meaningful grouping and construct a set of rules describing the interesting patterns discovered using the clustering algorithm proposed in Chapter 3. By incorporating transformation functions for performing data transformation, we can discover interesting patterns which cannot be found in the original data.

Currently, data from a wide range of application areas are stored in the database system [Elmasri and Navathe 1994]. Data such as the customer information, inventory records, or credit histories, etc. are usually stored as relational data whereas data such as the day-to-day transactions (e.g. the purchasing records, fund transfer records, etc) are usually stored as transaction data. Due to the database model for relational database systems, some of the attributes containing interesting patterns are not directly stored in the databases. If no transformation is taken, patterns inside such attributes not originally contained in a database cannot be discovered. For example, the "age" attribute is not always stored in relational database because it is a derived attribute [Elmasri and Navathe 1994]. With the reason that one's age can be determined from the current date and the date of birth of that person, an attribute "date of birth" is thus stored in the database systems instead of "age". Therefore, data mining process would be more effective if the derived attributes are transformed before applying data mining algorithms.

In addition to this, data transformation is also important for transaction data. For example, while it may be difficult to discover patterns between the start and end time of the subscribers, it may be an interesting pattern between the duration of a phone call for every transaction. If we consider only the attributes originally contained in the database, clusters that are described by a long distance phone call by subscribers cannot be discovered. To be able to discover such patterns, one must consider the new attribute of "duration". As another example, the actual amount of money a customer spent on any particular transactions may be irrelevant but the "average" over a certain period of time could have strong relationship with the income of the customer. The aim of data transformation is therefore to find some potentially useful attributes that are transformed from the original attributes in the database.

In addition to the handling of relational or transaction data separately, existing database systems always contain a mix of relational and transaction data. It is, therefore, important to handle both of them at the same time. For example, (i) a set of items bought by a customer in a purchase from a supermarket; (ii) an electronic fund transfer operation in a bank; or (iii) a phone call made by a customer of a telecommunication company. Specifically, the database system of the point-of-sales system in a supermarket stores not only the identity but also the price and the quantity of each item bought in each purchase. Another example is the electronic fund transfer system in a bank. The transaction database of such system maintains the operations carried out as well as, for instance, the accounts to be deposited and withdrawn in each fund transfer. In telecommunication system, each phone call is manifested in the form of a transaction record which consists of caller ID, time of call origination, and call duration, etc.

With the reason that many existing clustering techniques assume to perform their task in a single database table [MacQueen 1967; Ward 1963; Ng, Han 1994; Ester *et al.* 1996; Zhang, Ramakrishnan, and Livny 1996; Michaud 1997; and Huang 1997a, 1997b, 1998], it is not clear how well they are able to cope with data in a mix of multiple relational and transaction tables which are typically found in real-life database systems. Presumably, of course, when there are more than one table, the concept of *universal relation* can be used. A universal relation is an imaginary relation that can be used to represent the data constructed by logically joining all separate tables of a relational database [Ullman 1989]. The use of a *universal relation*, therefore, makes it possible for existing data mining systems (see, e.g. [Matheus, Chan, and Piatetsky-Shapiro 1993]) to deal with both transaction and relational data together if we assume that they are both stored in relational tables. But even if such assumption is valid, the construction of universal relations will very likely result in the introduction of redundant information that may decrease the effectiveness of many clustering algorithms for data mining [MacQueen 1967; Ward 1963; Ng, Han 1994; Ester *et al.* 1996; Zhang, Ramakrishnan, and Livny 1996; Michaud 1997; and Huang 1997a, 1997b, 1998]. Consider, for example, a database system in a retail store where each purchase corresponds to a record in a transaction database. Suppose there are 500 female customers and 500 male customers in the relational database. Suppose also that for all female customers, they made 20 purchases for a period of time and for all male customers, they made only 2 purchases for the same period of time. After applying the *universal relation* to the transaction and relational database, the number of records characterized by female customers would be $500 \times 20 = 10{,}000$ and that by male customers would be $500 \times 2 = 1{,}000$. For all the records (i.e. $10{,}000 + 1{,}000 = 11{,}000$), there would be more than 90% records characterized by female customers. Only less than 10% of records are characterized by male customers. As a consequence, the distribution of attribute *sex* in the table would be highly skewed. No matter how effective a data mining technique is, it is undoubtedly that interesting patterns are hardly to be extracted from such kind of data. This may further result in the discovery of low quality clusters. Instead of such skewed distribution between male and female customers, the original distribution should be 50% for both male and female customers (500 male customers and 500 female customers).

One of the reasons why many data mining techniques for clustering are not very effective to handle both transaction and relational data together is mainly due to the induction of redundant information when using *universal relation*. As a result, they cannot be used to discover clusters concerning, say, both the patterns about the background of a group of customers and transaction information during the purchase. In other words, a cluster characterized by male customers whose age over 20 always made purchases with average more than 1,000 dollars cannot be discovered if a data mining technique deals only with transaction records but not also with the background information of customers.

Taking into consideration the need to address these issues, we introduce here a problem formalism for data mining in database systems that collect both transaction and relational data. Based on this formalism, we present a new technique for data mining. Specifically, we define a set of transformation functions for different types of attributes in the databases of a database system to construct new relations. We then make use of the clustering algorithm introduced in Chapter 3 to uncover interesting patterns and discover meaningful grouping hidden in these transformed relations.

To test for effectiveness, we used a set of data collected in a real database system of point-of-sales system maintained by a retail and trade company. The database contains 10 to 50 attributes. This relational data has over 20,000 records in total. The database that contains the transaction data has over 350,000 purchase records that were collected over a two-year period. Experimental results show that the technique is able to discover clusters with interesting and useful descriptive rules for customer profiling and market segmentation. In particular, the relationship between customer background and purchase patterns discovered was found to be so valuable to the management of the marketing division that they expect significant cost saving in their marketing effort.

In the next section, we discuss the schema of database systems and show how the *universal relation* introduces redundant information which will affect the performance of data mining algorithms. In Section 5.2, we introduce a formalism of the problems involving the clustering tasks from both transaction and relational data in database systems. We then define, in Section 5.3, a collection of transformation functions that we use to deal with the problem of handling both types of data and to derive new attributes to facilitate the discovery of more interesting patterns. In Section 5.4, we illustrate how this technique can be used to handle both of transaction and relational data at the same time. We then finally discuss, in Section 5.5, the experimental results obtained by applying our technique to a real-life database of a point-of-sales system provided by a retail and trade corporation.

5.1 An Example

As discussed above, many data mining techniques are described under the assumption that there is only one relational table in a database. In the case when a database contains more than one table, the concept of a *universal relation* can presumably be used. A *universal relation* is an imaginary relation representing all the data constructed by logically joining separate tables of a relational database [Ullman 1989]. The use of such concept, therefore, makes possible for existing data mining systems (see, e.g., [Matheus, Chan, and Piatetsky-Shapiro 1993]) to be able to deal with more than one table. By constructing a *universal relation*, existing data mining systems can, theoretically, manipulate both transaction and relational data in database systems. However, the construction of *universal relations* will very likely result in the introduction of redundant information that will degrade the performance of the clustering process of many data mining algorithms [MacQueen 1967; Ward 1963; Ng, Han 1994; Ester *et al.* 1996; Zhang, Ramakrishnan, and Livny 1996; Michaud 1997; and Huang 1997a, 1997b, 1998].

We discuss how the *universal relation* can affect existing data mining algorithms by considering databases in a typical sales and inventory system. The major component of a sales and inventory system is a database system which allows customers to make purchase orders. A sales and inventory system provides customers with the current status of stocks so that they can make purchase order for any available products. In order to deal with the requirement that a customer can purchase one or more products in one transaction, such transactions are commonly maintained in two relational tables in the transaction database, namely, transaction-master and transaction-detail tables [Elmasri and Navathe 1994]. The transaction-master table keeps track of information concerning transactions as a whole, for instance, the transaction identifier, customer identifier, transaction information, and total amount of the transaction, etc. whereas the transaction-detail table stores data about the product purchased in each transaction such as the product identifier, number of product purchased, amount of a unit of product, discount of each product, and so on.

As an illustration, let us consider a simplified database in a sale and inventory system which is composed of CUSTOMER table, TRANSACTION_MASTER table, and TRANSACTION_DETAIL table. The CUSTOMER table belongs to the relational data whereas TRANSACTION_MASTER and TRANSACTION_DETAIL tables constitute the transaction database. Table 19(a-c) show records contained in these tables and Table 20 depicts the universal relation of this sample database.

CUSTOMER_ID	SEX	AGE	SALARY
001	M	37	35,000
002	F	23	20,000
003	F	31	25,000
004	M	25	23,000
005	M	42	40,000

(a) The CUSTOMER table.

TRANSACTION_ID	CUSTOMER_ID	DATE	TOT_AMOUNT
1000000001	003	4 Feb 98	15,000
1000000002	004	23 Feb 98	4,000
1000000003	001	2 Mar 98	5,000
1000000004	002	4 Mar 98	7,000
1000000005	002	26 Mar 98	2,000
1000000006	003	6 Apr 98	11,000
1000000007	005	15 Apr 98	6,000
1000000008	005	16 Apr 98	14,000
1000000009	001	26 Apr 98	12,000
1000000010	004	3 May 98	9,000

(b) The TRANSACTION_MASTER table.

TRANSACTION_ID	PRODUCT_ID	QUANTITY	UNIT_AMOUNT
1000000001	AA003	1	3,000
1000000001	DF341	1	2,000
1000000001	FG234	2	4,000
1000000001	WE651	1	2,000
1000000002	GE123	1	4,000
1000000003	VC324	3	1,500
1000000003	YT632	1	500
1000000004	HT932	2	3,500
1000000005	HT436	4	500
1000000006	BS458	1	8,000
1000000006	YU653	1	1,000
1000000006	MG348	2	1,000
1000000007	FS435	1	4,500
1000000007	UK234	1	1,500
1000000008	VJ456	1	3,000
1000000008	GR356	2	5,000
1000000008	LI964	1	1,000
1000000009	JU545	1	2,000
1000000009	FB324	2	5,000
1000000010	SR356	3	2,500

(c) The TRANSACTION_DETAIL table.

Table 19. A sample database of a sale and inventory system.

TRANSACTION_ID	PRODUCT_ID	QTY	UNIT_AMOUNT	CUSTOMER_ID	DATE	TOTAL_AMOUNT	SEX	AGE	SALARY
1000000001	AA003	1	3,000	003	4 Feb 98	15,000	F	31	25,000
1000000001	DF341	1	2,000	003	4 Feb 98	15,000	F	31	25,000
1000000001	FG234	2	4,000	003	4 Feb 98	15,000	F	31	25,000
1000000001	WE651	1	2,000	003	4 Feb 98	15,000	F	31	25,000
1000000002	GE123	1	4,000	004	23 Feb 98	4,000	M	25	23,000
1000000003	VC324	3	1,500	001	2 Mar 98	5,000	M	37	35,000
1000000003	YT632	1	500	001	2 Mar 98	5,000	M	37	35,000
1000000004	HT932	3	2,000	002	4 Mar 98	7,000	F	23	20,000
1000000005	HT436	4	500	002	26 Mar 98	2,000	F	23	20,000
1000000006	BS458	1	8,000	003	6 Apr 98	11,000	M	31	25,000
1000000006	YU653	1	1,000	003	6 Apr 98	11,000	M	31	25,000
1000000006	MG348	2	1,000	003	6 Apr 98	11,000	M	31	25,000
1000000007	FS435	1	4,500	005	15 Apr 98	6,000	M	42	40,000
1000000007	UK234	1	1,500	005	15 Apr 98	6,000	M	42	40,000
1000000008	VJ456	1	3,000	005	16 Apr 98	14,000	M	42	40,000
1000000008	GR356	2	5,000	005	16 Apr 98	14,000	M	42	40,000
1000000008	LI964	1	1,000	005	16 Apr 98	14,000	M	42	40,000
1000000009	JU545	1	2,000	001	26 Apr 98	12,000	M	37	35,000
1000000009	FB324	2	5,000	001	26 Apr 98	12,000	M	37	35,000
1000000010	SR356	3	2,500	004	3 May 98	9,000	M	25	23,000

Table 20. The universal relation of the sample database.

In the original data, there are 3 out of 5 customers (CUSTOM_ID = 001, 003 and 005) having salaries equal or greater than \$25,000. After constructing the universal relation tabulated in Table 20, there are 16 out of 20 records with salaries equal or greater than \$25,000. The percentage is changed from 60% (original data) to 80% (universal relation). Clearly, the percentage after forming the *universal relation* table is not equal to the actual data distribution. Since the distribution of data would directly affect the mean, mode, and probability distribution of data, etc., using the *universal relation* without any adjustment may make many data mining algorithms [MacQueen 1967; Ward 1963; Ng, Han 1994; Ester *et al.* 1996; Zhang, Ramakrishnan, and Livny 1996; Michaud 1997; and Huang 1997a, 1997b, 1998] ineffective and may result in providing inaccurate input to these algorithms. Taking this into consideration, we present a problem formalism and describe a technique for data mining based on this formalism in the next section.

5.2 Handling Both Transaction and Relational Data

Let $A_{i1}, A_{i2}, \ldots, A_{in_i}, i = 1,2,\ldots,I$ be the attributes of the real-world entities represented by the relational tables, $R_i, i = 1,2,\ldots,I$, respectively. Let the domain of

$A_{ik}, i = 1,2,\ldots,I$ and $k = 1,2,\ldots,K_i$, be represented by $dom(A_{ik}) = \{a_{ik}^{(1)}, a_{ik}^{(2)}, \ldots, a_{ik}^{(m_{ik})}\}, i = 1,2,\ldots,I, k = 1,2,\ldots,K_i$. In other words, $R_i \subseteq dom(A_{i1}) \times dom(A_{i2}) \times \cdots \times dom(A_{iK_i})$. For any R_i, we use $\mathcal{A}_{R_i}$ to denote the set of attributes of R_i, that is, $\mathcal{A}_{R_i} = \{A_{i1}, A_{i2}, \ldots, A_{iK_i}\}$. The primary key of R_i, which is composed of one or more attribute-value pairs and is associated with each n-tuple in a relation is represented as $\mathcal{K}_i \subseteq \{A_{i1}, A_{i2}, \ldots, A_{iK_i}\}$.

Let us also denote a set of transaction records as $T_j, j = 1,2,\ldots,J$, where each T_j is characterized by a set of attributes denoted by $A_{j1}, A_{j2}, \ldots, A_{jL_j}$ and has a unique transaction identifier TID_j. In other words, $T_j \subseteq TID_j \times dom(A_{j1}) \times dom(A_{j2}) \times \cdots \times dom(A_{jL_j})$. In a database system, there exists some one-to-many relationships [Elmasri and Navathe 1994] between the records in $R_i, i = 1,2,\ldots,I$ and those in $T_j\, j = 1,2,\ldots,J$.

For instance, a business transaction processing system may contain: (i) a set of relational tables containing background information about customers and (ii) a transaction database containing details of each transaction made by customers such as transaction date, transaction time, and amount purchased. The relational data are related to the transaction data in some one-to-many relationship such that we can find $\mathcal{K}_i$, which is the primary key of R_i, in $\{A_{j1}, A_{j2}, \ldots, A_{jL_j}\}$, that can be used as foreign key to provide reference to the corresponding n-tuple in $R_i, i = 1,2,\ldots,I$.

Given R_i and T_j, to deal with both relational and transaction data at the same time and to consider additional attributes not originally in the databases, we propose to use the concept of transformation functions defined on the original attributes in R_i and T_j. Let $f_1, f_2, \ldots, f_p$ be a set of transformation functions such that

$$f_p : A_{p1} \times A_{p2} \times \cdots \times A_{pr_p} \rightarrow A'_p, p = 1,2,\ldots,P$$

where $r_p \geq 1$

$$\text{and } A_{pu} \in \left(\bigcup_{i=1}^{I} \mathcal{A}_{R_i}\right) \cup \left(\bigcup_{j=1}^{J} \mathcal{A}_{T_j}\right), u = 1,2,\ldots,r_p$$

An example of a transformation function, say f_1, is to discretize a numeric attribute *Working Experience* into the following four categories: *Working Experience* ≤ 2, 3 ≤ *Working Experience* ≤ 5, 6 ≤ *Working Experience* ≤ 10, *Working Experience* ≤ 11. f_1 will produce a new attribute *Working Experience'* by the following mapping:

$$Working\ Experience' = \begin{cases} 1 & \text{if } Working\ Experience \le 2 \\ 2 & \text{if } 3 \le Working\ Experience \le 5 \\ 3 & \text{if } 6 \le Working\ Experience \le 10 \\ 4 & \text{if } Working\ Experience \le 11 \end{cases}$$

Another example of a transformation function, say f_2, is to calculate the average price of goods purchased by each customer in a transaction of a point-of-sale system. For each customer with identifier α_{iu}, f_2 will generate the transformed attribute

$$Price' = \frac{\sum_{t \in \sigma_{\mathcal{K}_i = \alpha_{iu}}(T_j)} t[\ Price]}{|\sigma_{\mathcal{K}_i = \alpha_{iu}}(T_j)|}$$

where σ denotes the SELECT operation in *relational algebra* and |*S*| denotes the *cardinality* of set *S* [Elmasri and Navathe 1994].

We can construct a new relation R' which contains both the original attributes in R_i and T_j and transformed attributes obtained by applying appropriate transformation functions. Let R' be composed of attributes $A_1', A_2', \ldots, A_n'$, that is $R' \subseteq dom(A_1') \times dom(A_2') \times \cdots \times dom(A_u') \cdots \times dom(A_n')$ where A_u', $u = 1,2,\ldots,n$ can be any attribute in $R_i, i = 1,2,\ldots,I$ or $T_j, j = 1,2,\ldots,J$ or any transformed attribute. In other words, $A_u' \in \left(\bigcup_{i=1}^{I} \mathcal{A}_{R_i}\right) \cup \left(\bigcup_{j=1}^{J} \mathcal{A}_{T_{ij}}\right) \cup \left(\bigcup_{p=1}^{P} f_p(A_{p1,\ldots,}A_{pr_p})\right)$. Instead of performing data mining on the original R_i and T_j, we perform data mining on R'.

5.3 Transformation Functions

Given a database system, the following functions [Au 1998] can be performed. They include the logical, arithmetic, partition, and discretization functions. Depending on the attribute type, one or more of them can be applied to it.

5.3.1 Logical Functions

Logical functions are composed of a combination of logical operators such as NOT, AND, OR, etc. A logical function can take one or more attributes as argument. Let $f_1, f_2, \ldots, f_n$ be a set of functions such that:

$$f_j(a_1,a_2,...,a_r) = (a_1 = c_{j1}) \oplus (a_2 = c_{j2}) \otimes \cdots \Theta(a_r = c_{jr}), j = 1,2,...,n$$

where $a_j \in dom(A_i), c_i \in dom(A_i), A_i \in \left(\bigcup_{i=1}^{I} A_{R_i}\right)\bigcup\left(\bigcup_{j=1}^{J} A_{T_j}\right), i = 1,2,...,r$

and $\oplus, \otimes, \cdots, \Theta \in \{\text{AND}, \text{OR}, \text{NOT}, \text{XOR}, \text{NAND}, \text{NOR}\}$

A generic way of utilizing these functions is to construct a logical function, f, defined in terms of $f_1, f_2, \ldots, f_n$ as follows:

$$f(a_1,a_2,\cdots,a_r) = \begin{cases} 1 & \text{if } f_1(a_1,a_2,\cdots,a_r) = \text{true} \\ 2 & \text{if } f_2(a_1,a_2,\cdots,a_r) = \text{true} \\ . & \\ . & \\ . & \\ n & \text{if } f_n(a_1,a_2,\cdots,a_r) = \text{true} \end{cases}$$

where $a_j \in dom(A_i), A_i \in dom(A_i), A_i \in \left(\bigcup_{i=1}^{I} A_{R_i}\right)\bigcup\left(\bigcup_{j=1}^{J} A_{T_j}\right), i = 1,2,...,r$

In case that the value of any attribute, A_i, i=1, 2, …, r, of a record is missing or unknown, the logical function f will produce an unknown value as output.

Consider, for example, the attribute *Sex* and *Occupation* in a given database, we can construct a new attribute represent both the sex and occupation. The *sex_occupation* function is used to determine the value of the new attribute based on the values of *Sex* and *Occupation*. An implementation of the *sex_occupation* function is shown in Fig. 13.

```
int sex_occupation (Sex, Occupation)
begin
    if $Sex = female$ and $Occupation = engineering$ then
        return 1;
    elseif $Sex = female$ and $Occupation = finance$ then
        return 2;
    elseif $Sex = female$ and $Occupation = banking$ then
        return 3;
    elseif $Sex = male$ and $Occupation = engineering$ then
        return 4;
    elseif $Sex = male$ and $Occupation = finance$ then
        return 5;
    elseif $Sex = male$ and $Occupation = banking$ then
        return 6;
end
```

Figure 13. An example of logical functions.

5.3.2 Arithmetic Functions

Arithmetic functions can involve addition, subtraction, multiplication, and division. An arithmetic function takes a set of attributes as argument and produces an attribute of type real or integer. The arithmetic function *f* is defined as

$$f\ (a_1, a_2, \ldots, a_r) = a_1 \oplus a_2 \otimes \cdots \Theta a_r$$

where $a_j \in dom(A_i), A_i \in dom(A_i), A_i \in \left(\bigcup_{i=1}^{I} A_{R_i}\right) \bigcup \left(\bigcup_{j=1}^{J} A_{T_j}\right), i = 1,2,\ldots,r$

and $\oplus, \otimes, \cdots, \Theta \in \{+, -, \times, \div\}$

In case that the value of any attribute, A_i, i=1, 2, ..., r, of a record is missing or unknown, the arithmetic function will produce an unknown value as output.

An example of arithmetic functions is the calculation of bonus of a staff. It can be defined as follows (Fig. 14)

```
real bonus(Amount)
begin
        real remaining, tot_bonus;
        if amount > 40,000 then
        begin
                remaining = amount – 40,000;
                tot_bonus = 15,000 × 3% + 25,000 × 5% + remaining × 10%;
        end
        elseif Amount > 15,000 then
        begin
                remaining = amount – 15,000;
                tot_bonus = 15,000 × 3% + remaining × 5%;
        end
        else
                tot_bonus = amount × 3%;
        return tot_bonus;
end
```

Figure 14. An example of arithmetic functions.

5.3.3 Partition Functions

Partition functions extract a specific portion from a given attribute. Let the given attribute A be a string of s alphabets. For any $a \in dom(A)$, we use $a[i]$ to denote the i^{th} alphabet of a. The partition function f is defined as

$$f(a) = a[l]a[l+1]\cdots a[u]$$

where $a \in dom(A) \in \left(\bigcup_{i=1}^{I} A_{R_i}\right) \bigcup \left(\bigcup_{j=1}^{J} A_{T_j}\right)$

and $1 \le l \le u \le s$

In case that the value of attribute A of a record is missing or unknown, the partition function f will produce an unknown value as output.

As an illustration, the call number of a book in library can be in the form "XX9999.99.X99X99" where the first or the first two digits denote the category of the book. To obtain information about the category the book, we can apply a partition function. The *category* function illustrated in Fig. 15 extracts the category code from a given book. *CallNumber*[1..2] denotes the first two digits of *CallNumber*.

```
code category(CallNumber)
begin
    return CallNumber[1..2];
end
```

Figure 15. An example of partition functions.

5.3.4 Discretization Functions

Discretization functions discretize any numeric attribute into finite intervals. Let f be the discretization function such that there are r intervals. We use u_i to denote the upper limit of i^{th} interval for i=1, 2, ..., r-1. The f is defined as

$$f(a) = \begin{cases} 1 & \text{if } a \le u_1 \\ 2 & \text{if } u_1 < a \le u_2 \\ . & \\ . & \\ . & \\ r-1 & \text{if } u_{r-2} < a \le u_{r-1} \\ r & \text{if } a > u_{r-1} \end{cases}$$

where $a \in dom(A) \in \left(\bigcup_{i=1}^{I} A_{R_i} \right) \bigcup \left(\bigcup_{j=1}^{J} A_{T_j} \right)$

In case that the value of attribute A of a record is missing or unknown, the discretization function f will produce an unknown value as output.

The boundaries of the intervals can be specified by users or determined by computer algorithm (see, e.g., [Chiu, Cheung, and Wong 1990; Cheung, Wong, and Chan 1995]) automatically. One of the commonly used algorithms is to discretize the attribute into equal intervals. Another popular algorithm is to discretize the attribute into intervals such that the number of records in each interval is the same. As a result, each record has equal probability to fall into any interval.

Consider, for instance, the attribute *Age* in the database, we want to categorize it into youngster, adult, and elder by the use of the *age* function in Fig. 16:

```
int age (Age)
begin
    if Age < 20 then
        return 1; // youngster
    elseif Age < 55 then
        return 2; // adult
    else
        return 3; // elder
end
```

Figure 16. An example of discretization functions.

5.3.5 Examples

A database system consists of attributes that take on different types of values such as categories/types, code/ID, name, address, phone number, quantity/amount, date, time, and comments/remarks. Attributes of type categories/types can take on finite possible values and they are not used as identifiers belong to this attribute type. An example is marital status in which single, married, divorced, and widowed are the only possible values. We can apply logical functions to this attribute type. For instance, a new attribute can be constructed by the logical function that "*marital status = single*" and "*sex = female*".

There are attributes that take on a string of alphanumeric such as ID numbers. These *codes/Ids* are served as identifiers. When the *code/ID* is composed of several components, we can extract these components out of the *code/ID* using partition functions. For example, we can get the branch code from a saving account number in Hong Kong. Nevertheless, if the *code/ID* is assigned from flat name space such as Staff ID Number, there is no available transformation function. And such attribute should be ignored in data mining.

Furthermore, there are attributes about names or special terms, for example, staff name, and company name, etc. We can apply partition functions to this type of attributes. As an example, we can extract first name and last name for a specific person. Another example is that we can determine the subsidiaries and mother company for a specific company.

There are also attributes that contain strings of words concerning with addresses such as company address, and home address, etc. We can apply partition functions which may be followed by the logical functions to these attributes. For instance, we can extract district, city, country, and postal code from a specific address.

Attributes such as phone numbers can be assigned in form of hierarchical structure, we are able to extract corresponding code from a specific phone number by partition functions and can treat the extracted portion as *categories/types*. As an example, we can extract the first digit from a phone number in Hong Kong and we can identify the type of phone by using this digit, e.g. 7 for pagers, 9 for mobile phones, and 3 for Internet Services Provider. However, if phone number is assigned in flat name space, no transformation function is available.

There are typical numeric attributes whose values are real or integer numbers, for instance, number of accounts, total amounts to be paid, and discount rate, etc. We can apply arithmetic functions, discretization functions to this kind of attributes. For example, we can discretize the attribute age into youngster, adult, and elder.

We can apply arithmetic functions and discretization functions to attributes concerning with date or time. For instance, we can discretize a specific date into *weekday/weekend*, *working day/holiday*, *Christmas period* or not, *New Year* period or not, and so on.

5.4 An Illustrative Example

In this section, we illustrate the technique we have proposed in this chapter to handle both transaction and relational database by giving a concrete example. Assume that there is a synthetic database containing customer background records and the courier services transaction records of customers using the courier services provided by a courier services corporation. In other words, the database consists of the following two relational tables.

DELIVERY (TID, CUST_ID, DATE, WEIGHT, SOURCE_COUNTRY, DESTINATION_COUNTRY)

CUSTOMER (CUST_ID, ACCT_NUM, CUST_NAME, SEX, MARITAL STATUS, DATE_BIRTH)

These relational tables and the types of their attributes are shown in Table 21(a-b) and Table 22(a-b) respectively.

TID	CUST_ID	DATE	WEIGHT	SOURCE_COUNTRY	DESTINATION_COUNTRY
30000001	214	1 Feb 99	2.5	China	US
30000002	215	1 Feb 99	3.4	US	UK
30000003	214	1 Feb 99	2.4	UK	China
30000004	218	1 Feb 99	5.5	Japan	Italy
30000005	215	2 Feb 99	8.2	Canada	France
30000006	215	2 Feb 99	8.1	China	Canada
30000007	217	2 Feb 99	10.5	US	Australia
30000008	214	2 Feb 99	1.5	Japan	US
30000009	218	2 Feb 99	6.2	Italy	Australia
30000010	211	2 Feb 99	5.1	UK	China
30000011	217	3 Feb 99	5.9	France	UK
30000012	218	3 Feb 99	1.2	China	Japan
30000013	214	3 Feb 99	6.5	Australia	Japan
30000014	213	3 Feb 99	8.2	Italy	China
30000015	215	3 Feb 99	20.1	France	Canada
30000016	215	3 Feb 99	5.3	Australia	US
30000017	218	3 Feb 99	1.5	China	Australia
30000018	214	3 Feb 99	9.2	UK	Australia
30000019	216	4 Feb 99	7.1	Canada	Japan
30000020	213	4 Feb 99	8.8	US	Australia
30000021	216	4 Feb 99	1.1	Japan	China
30000022	218	4 Feb 99	6.1	China	UK
30000023	214	4 Feb 99	9.4	France	US
30000024	216	4 Feb 99	15.3	Italy	Canada
30000025	214	5 Feb 99	1.9	UK	France
30000026	211	5 Feb 99	18.5	China	Italy
30000027	212	5 Feb 99	8.7	China	US
30000028	218	5 Feb 99	1.6	US	Japan
30000029	214	5 Feb 99	6.1	Canada	UK
30000030	219	5 Feb 99	8.1	China	Italy

(a) The DELIVERY table.

CUST_ID	ACCT_NUM	CUST_NAME	SEX	MARITAL_STATUS	DATE_BIRTH
211	16551634	Peter Chan	M	Single	21 Jan 65
212	46851615	Jack Choi	M	Married	12 Aug 68
213	48948956	Ann Lee	F	Widowed	27 Nov 73
214	61588515	Joyce Lam	F	Married	04 Feb 59
215	45688486	Chris Ho	M	Divorced	16 Apr 75
216	89884436	Mary Yeung	F	Married	29 Dec 66
217	15677624	Albert Ng	M	Single	13 May 71
218	58445135	May Wo	F	Single	23 Jun 70
219	56898954	Angle Tong	F	Married	30 Sep 69

(b) The CUSTOMER table

Table 21. Relational tables in the illustrative database.

Attribute	Type	Description
TID	Code/ID	Transaction Identifier
CUST_ID	Code/ID	Customer ID No.
DATE	Date	Date of delivery
WEIGHT	Quantity/Amount	Weight of parcel
SOURCE_COUNTRY	Categories/Types	Source country
DESTINATION_COUNTRY	Categories/Types	Destination country

(a) The DELIVERY table.

Attribute	Type	Description
CUST_ID	Code/ID	Customer ID No.
ACCT_NUM	Code/ID	Account Number
CUST_NAME	Name	Customer Name
SEX	Categories/Types	Sex
MARITAL_STATUS	Categories/Types	Marital Status
DATE_BIRTH	Date	Date of Birth

(b) The CUSTOMER table.

Table 22. Attributes and their types of the (a) DELIVERY and (b) CUSOMTER tables in the illustrative database.

We intend to construct a transformed relation R(ACCT_TYPE, NUM_WEEKDAY, NUM_WEEKEND, REGION_ASIA_EUROPE, WEIGHT) using the transformation functions described in Section 5.3.

5.4.1 Transformation of ACCT_TYPE

Assume that the last digit of attribute CUSTOMER[ACCT_NUM] denotes the account type. For example, if the last digit is 4, it is a saving account. Similarly, value 5 denotes current account and value 6 denotes checkbook account. The transformation function $f_{\text{ACCT\_TYPE}}$ can be define as:

$$f_{\text{ACCT\_TYPE}}(s) = \begin{cases} 1 & \text{if } last\_digit\_of(s) = 4 \\ 2 & \text{if } last\_digit\_of(s) = 5 \\ 3 & \text{if } last\_digit\_of(s) = 6 \end{cases}$$

where *last_digit_of*(s) is a function that returns the last digit of string s. The transformed attribute ACCT_TYPE is produced by the extraction a part from $f_{\text{ACCT\_TYPE}}$(BACKGROUND[ACCT_NUM]) for every n-tuple in CUSTOMER. This is achieved by applying partition function described in Section 5.3.3.

5.4.2 Transformation of NUM_WEEKDAY and NUM_WEEKEND

In a typical database, an attribute that is always stored is the DATE field. However, such attribute is not meaningful enough for data mining technique if no transformation is applied. By discretizing the date of the delivery into weekday and weekend, more interesting patterns may be discovered. The transformation function $f_{\text{DATE\_TYPE}}$ is defined as:

$$f_{\text{DATE\_TYPE}}(d) = \begin{cases} 1 & \text{if } d \in \{\text{Mon}, \text{Tue}, \text{Wed}, \text{Thu}, \text{Fri}\} \\ 2 & \text{if } d \in \{\text{Sat}, \text{Sun}\} \end{cases}$$

where 1 and 2 represent weekday and weekend respectively. Actually $f_{\text{DATE\_TYPE}}$ is a discretization functions described in Section 5.3.4.

In order to have a more meaningful source data, the number of delivery placed on weekday and weekend are generated using aggregate function, $f_{\text{NUM\_WEEKDAY}}$ and $f_{\text{NUM\_WEEKEND}}$ which are defined as follows:

$$f_{\text{NUM\_WEEKDAY}}(id) =_{id} \xi_{\text{COUNT TID}}(\sigma_{\text{CUST\_ID}=id \text{ AND } f_{\text{DATE\_TYPE}}(\text{DELIVERY[DATE]})=1}\text{DELIVERY})$$

and $f_{\text{NUM\_WEEKEND}}(id) =_{id} \xi_{\text{COUNT TID}}(\sigma_{\text{CUST\_ID}=id \text{ AND } f_{\text{DATE\_TYPE}}(\text{DELIVERY[DATE]})=2}\text{DELIVERY})$

where ξ denotes the *aggregate function* in relational algebra [Elmasri and Navathe 1994]. The transformed attributed NUM_WEEKDAY and NUM_WEEKEND are generated by $f_{\text{NUM\_WEEKDAY}}(id)$ and $f_{\text{NUM\_WEEKEND}}(id)$ for all $id \in \pi_{\text{CUST\_ID}}(\text{CUSTOMER})$ respectively.

5.4.3 Transformation of REGION_ASIA_EURORE

The SOURCE_COUNTRY and DESTINATION_COUNTRY of a delivery can be grouped into different geographical regions for the discovery of more interesting patterns. Such grouping is performed by the transformation function f_{REGION}. It is defined as:

$$f_{\text{REGION}}(n) = \begin{cases} 1 & \text{if } n = \text{China or Japan or ... or Australia} \\ 2 & \text{if } n = \text{UK or France or ... or Italy} \\ 3 & \text{if } n = \text{US or Canada} \end{cases}$$

where 1 denotes Asia, 2 denotes Europe, and 3 denotes North American. f_{REGION} is an example of the logical function described in Section 5.3.1. Since the relationship between source and destination region may contain more useful information, another logical function is then applied to the result of $f_{\text{REGION}}(n)$. Let s be the result of f_{REGION}(DELIVERY[SOURCE_COUNTRY]) and d be the result of f_{REGION}(DELIVERY[DESTINATION_COUNTRY]). The transformation function $f_{\text{REGION\_FROMTO}}(s,d)$ is defined as:

$$f_{\text{REGION\_FROMTO}}(s,d) = \begin{cases} 1 & \text{if } s = 1 \text{ and } d = 2 \\ 2 & \text{if } s = 1 \text{ and } d = 3 \\ 3 & \text{if } s = 2 \text{ and } d = 1 \\ 4 & \text{if } s = 2 \text{ and } d = 3 \\ 5 & \text{if } s = 3 \text{ and } d = 1 \\ 6 & \text{if } s = 3 \text{ and } d = 2 \end{cases}$$

Finally, the REGION_ASIA_EUROPE are generated by aggregate function, $f_{\text{REGION\_ASIA\_EUROPE}}$ which is given as:

$$f_{\text{REGION\_ASIA\_EURPOE}}(id) = {}_{id}\, \xi_{\text{COUNT TID}}(\sigma_{\text{CUST\_ID}=id \text{ AND } f_{\text{REGION\_FORMTO}}(s,d)=1}\,\text{DELIVERY})$$

where ξ denotes the *aggregate function* in relational algebra [Elmasri and Navathe 1994]. Besides the transformed attribute REGION_ASIA_EUPOPE, other attributes can also be transformed such as REGION_ASIA_NORTHAMERICA, REGION_EUROPE_ASIA, etc. in the similar way.

5.4.4 Transformation of WEIGHT

In order to compute the average duration of those deliveries made by customers, we make use of another transformation function $f_{\text{AVG\_WEIGTH}}$ such that

$$f_{\text{AVG\_WEIGHT}}(id) = \frac{\sum_{t \in \sigma_{\text{CUST\_ID}=id}(\text{DELIVERY})} t[\text{WEIGHT}])}{|\sigma_{\text{CUST\_ID}=id}(\text{DELIVERY})|}$$

where σ denotes SELECT operation of relational algebra. The function $f_{\text{AVG\_WEIGHT}}$ is an example of the arithmetic functions described in Section 5.3.2. We can then utilize another transformation function f_{WEIGHT} which is given by

$$f_{\text{WEIGHT}}(x) = \begin{cases} 1 & \text{if } x \le 5 \\ 2 & \text{if } 5 < x \le 50 \\ 3 & \text{if } x > 50 \end{cases}$$

The function f_{WEIGHT} is, in fact, an example of the discretization functions described in Section 5.3.4. The transformed attribute WEIGHT is computed by $f_{\text{WEIGHT}}(f_{\text{AVG\_WEIGHT}}(id))$ for all $id \in \pi_{\text{CUST\_ID}}(\text{CUSTOMER})$ where π denotes PROJECT operation of relational algebra. The transformed relation together with some representative *n*-tuples are shown in Table 23.

ACCT_TYPE	N_WEEK DAY	N_WEEK END	REGION_ASIA_ EUROPE	WEIGHT
1	0	2	0	0
2	2	1	0	1
3	2	0	1	1
2	1	0	2	2
3	1	2	2	0
3	2	2	1	0
1	0	1	0	2
2	0	1	2	2
1	1	1	2	1

Table 23. The transformed relation of the illustrative database.

After data transformation, we can then apply the clustering algorithm introduced in Chapter 3 to the transformed relation to discover clusters and interesting patterns.

5.5 Experimental Results

With transformation, we used our clustering algorithm to mine real data taken from a point-of-sale system provided by retail and trading corporation. The database contains relational data of over 20,000 records and 20 relational tables and each of which consists of 10 to 50 attributes. The other database, the one that contains transaction data has over 350,000 transaction records that were collected over two-year period. These sales records are concerning the purchasing of either personal or company customers from a specific location of shop.

The relational database includes a PURCHASE table, a CUSTOMER table, and a DISCOUNT table. The PURCHASE table which contains transaction records of purchases made by customers over a two-year period has over 350,000 purchase records. The CUSTOMER table consists of the background information of about 20,000 customers. And the DISCOUNT table stores information about the discount rating determined by the marketing department for customers. Table 24(a-c) shows some of the attributes and their types of the PURCHASE, CUSTOMER, and DISCOUNT tables.

The corporation is interested in how the background of customers affects their purchasing patterns. The direct application of data mining techniques to the *universal relation* of the database may result in the neglect of some interesting patterns. For instance, cluster is characterized by customer whose occupation is marketing and finance with salary earned more than 30,000 purchase in large amount in December. In order to reveal such kind of knowledge, we have to apply appropriate transformation functions to the database before the clustering task.

Attribute	**Type**	**Description**
TID	Code/ID	Transaction Identifier
CUST_ID	Code/ID	Customer ID No.
Date	Date	Date of transaction
TradeID	Code/ID	Type of trade
TradeCode	Code/ID	Trade code
SalesmanID	Code/ID	Salesman ID
PaymentCode	Code/ID	Type of payment
ShopLoc	Code/ID	Shop Location
Amount	Quantity	Total amount of purchase

(a) The PURCHASE table.

Attribute	Type	Description
CUST_ID	Code/ID	Customer ID No.
CUST_NAME	Name	Customer Name
SEX	Categories/Types	Sex
M_STATUS	Categories/Types	Marital Status
DATE_BIRTH	Date	Date of Birth
TEL_NO	Phone No.	Telephone Number
ADDRESS	Address	Address
PAY_TYPE	Categories/Types	Payment Type
SALARY	Quantity	Monthly Salary
EDU_LV	Categories/Types	Education Level
OCCUP	Categories/Types	Occupation
CO_NAME	Name	Company Name
CO_TEL_NO	Phone No.	Company Telephone Number
CO_ADDRESS	Address	Company Address
DATE_REG	Date	Date becoming customer
ACC_NUM	Code/ID	Account No.
NO_OF_EMP	Quantity/Amount	No. of employees of company
INDUSTRY	Categories/Types	Industry of company

(b) The CUSTOMER table.

Attribute	Type	Description
CUST_ID	Code/ID	Customer ID No.
DIS_RATE	Quantity/Amount	Discount rate
DIS_TYPE	Categories/Types	Discount type

(c) The DISCOUNT table.

Table 24. Some of the attributes and their types of the (a) PURCHASE, (b) CUSTOMER, and (c) DISCOUNT tables.

We create a new relation from which clusters to be discovered. The relation is constructed by summarizing the PURCHASE table and then joining the transformed PURCHASE table with the CUSTOMER and the DISOCUNT tables. Consequently, each record represents a customer in the transformed relation. This relation is aimed at discovering knowledge of how the background of customers affects their purchase pattern. An example of the discovered pattern from a cluster is "Female customer aged between 30 to 40 with high purchase amount in the 2nd quarter of a year". The transformed relation is given in Table 25.

Attribute	Transformation Functions	Description
SEX	Nil	Sex
M_STATUS	Nil	Marital Status
AGE	Discretization and Arithmetic	Age
DISTRICT	Partition	District
SALARY	Discretization	Monthly Salary
EDU_LV	Logical	Education Level
OCCUP	Logical	Occupation
REG_YEAR	Discretization and Arithmetic	Number of year registered
ACCT_TYPE	Partition	Account type
COM_SIZE	Discretization	Company size
INDUSTRY	Nil	Primary industry
PAY_TYPE	Nil	Payment type
DIS_TYPE	Nil	Discount type
AMT_JAN	Discretization and Arithmetic	Average purchase amount in January
AMT_FEB	Discretization and Arithmetic	Average purchase amount in February
AMT_MAR	Discretization and Arithmetic	Average purchase amount in March
AMT_APR	Discretization and Arithmetic	Average purchase amount in April
AMT_MAY	Discretization and Arithmetic	Average purchase amount in May
AMT_JUN	Discretization and Arithmetic	Average purchase amount in June
AMT_JUL	Discretization and Arithmetic	Average purchase amount in July
AMT_AUG	Discretization and Arithmetic	Average purchase amount in August
AMT_SEP	Discretization and Arithmetic	Average purchase amount in September
AMT_OCT	Discretization and Arithmetic	Average purchase amount in October
AMT_NOV	Discretization and Arithmetic	Average purchase amount in November
AMT_DEC	Discretization and Arithmetic	Average purchase amount in December
AMT_Q11	Discretization and Arithmetic	Average purchase amount in 1st quarter of 1st year
AMT_Q21	Discretization and Arithmetic	Average purchase amount in 2nd quarter of 1st year
AMT_Q31	Discretization and Arithmetic	Average purchase amount in 3rd quarter of 1st year
AMT_Q41	Discretization and Arithmetic	Average purchase amount in 4th quarter of 1st year
AMT_Q12	Discretization and Arithmetic	Average purchase amount in 1st quarter of 2nd year
AMT_Q22	Discretization and Arithmetic	Average purchase amount in 2nd quarter of 2nd year
AMT_Q32	Discretization and Arithmetic	Average purchase amount in 3rd quarter of 2nd year
AMT_Q42	Discretization and Arithmetic	Average purchase amount in 4th quarter of 2nd year
AMT_YR1	Discretization and Arithmetic	Average purchase amount in 1st year
AMT_YR2	Discretization and Arithmetic	Average purchase amount in 2nd year

Table 25. The transformed relation.

After data transformation, we can apply the clustering algorithm described in Chapter 3 to the transformed relation to discover interesting patterns inside each cluster. The grouping of customers discovered from the database is regarded as extremely valuable by the management of the corporation. For example, there is a cluster that is characterized by male customers whose age is between 40 and 50 with tertiary education and with medium average purchase amount in February and September. There is another cluster that is characterized by married female customers working in finance area having medium average purchase amount in the first year and large average amount in the second year. In addition, there is also a cluster that is characterized by male customers working in engineering industry with secondary level education having large average purchase amount in the 4th quarter of 1st year and medium average purchase amount in the 2nd quarter of 2nd year. All the described clusters above contain some transformed attributes (e.g. average amount in February, average amount in 1st year, etc.). If no data transformation is taken, it is obvious that such patterns would not have been discovered. Furthermore, the ability of our technique to discover rules for customer profiling and market segmentation has significant importance to marketing efforts of the corporation.

In this chapter, we discussed the problem of discovering interesting clusters in database systems containing both transaction and relational data. We presented a formalism of this problem and proposed a solution for it. Specifically, we defined a set of transformation functions for different types of attributes in both transaction and relational data. Based on these functions, we constructed a set of new relations to discover clusters and interesting patterns hidden in these relations using the algorithm proposed in Chapter 3. By incorporating functions for performing data transformation, our approach is able to handle both transaction and relational data and discover interesting patterns which cannot be found in the original data. We also illustrated how to use the proposed technique to handle both of transaction and relational data. By transforming appropriate attributes, we showed that our technique is able to discover meaningful grouping of records and explicit represents the discovered knowledge in rules in a real-life database of a point-of-sales system provided by a retail and trade corporation. The clusters and rules discovered from this database are regarded as extremely valuable by the management of the marketing division of the corporation.

Chapter 6

Conclusion and Future Work

6.1 Summary

In this thesis, we described a new clustering algorithm based on the use of a simple genetic algorithm. A particular grouping of database records is encoded in a chromosome so that each gene represents a cluster label. Once different groupings are generated, the most interesting ones are determined by means of an evolutionary process guided by a fitness function. This fitness function is defined in terms of a probabilistic similarity measure and can be interpreted as an objective measure of interestingness of the rules that characterize the groupings encoded in a particular chromosome. Through maximizing such fitness function, the intra-class similarity is maximized and, at the same time, the inter-class similarity is minimized. The most interesting grouping of records can be evolved by the operations of selection, crossover, and mutation.

To evaluate the performance of our proposed clustering approach, we used many sets of real life data in our experimentation. We compared the performance of our clustering algorithm with popular clustering algorithms developed by artificial intelligence, pattern recognition and data mining researchers. They are (i) COBWEB; (ii) AUTOCLASS; (iii) *k*-means; (iv) *k*-modes; and (v) Condorcet. The results were evaluated by accuracy of different approaches by defining the percentage of records that are grouped in the same way as the original data. The higher the accuracy of the evaluated algorithm, the better is its performance. Our algorithm performed the best consistently in all real-life and simulated data. In addition, our clustering algorithm is also able to discover a set of if-then rules describing the patterns underlying all discovered clusters. For further testing, we also used simulated data. We successfully discovered all the patterns that are intentionally generated in simulated data. For the user's understanding, the interestingness of each discovered rule is then represented in terms of *support* and *confidence* values. From all the experimental results, they showed that the proposed approach was able to handle real world problems more effectively when compared to existing algorithms in all cases.

In summary, the clustering algorithm proposed has a number of advantages. Unlike many of the existing clustering algorithms for data mining, which are developed mainly to deal with spatial databases containing only continuous-valued attributes and the number of dimensions is limited, our algorithm is able to perform its task in both continuous- and discrete-valued data. Our algorithm, in turn, is suitable to work in typical databases containing both transaction and relational data. Also, our clustering algorithm does not employ a distance measure for similarity so it is better to deal with data that are noisy, incomplete, inconsistent and contain missing values. In addition to the discovery of non-overlapping clusters in many of the existing clustering algorithms, our algorithm is able to discover records that belong to one or more clusters. In other words, the boundaries of clusters allow overlapping. Furthermore, the mined knowledge in form of if-then rules is provided in our algorithm. This non-black-box approach makes it possible for the patterns underlying the databases to be made explicit. For the time complexity, our clustering algorithm is comparable to that of *k*-means and AUTOCLASS.

6.2 Future Work

For future work, the proposed clustering algorithm to deal with fuzziness will be enhanced [Chung and Chan]. This includes (i) the handling of fuzzy boundaries and (ii) the handling of fuzzy data. Instead of assigning a record to one or more clusters in existing clustering algorithms, some techniques can be developed to make it possible for each record to have a degree of membership with respect to each cluster [Zadeh 1965]. This can better handle real world problems for clustering algorithm equipped with fuzziness as many of these problems contain uncertainty, imprecision and vagueness.

In addition to the use of memberships for fuzzy boundary, clustering algorithms for data mining tasks would be more effective in dealing with more sophisticated problems if they could handle fuzzy data. In fact, data in many complicated problems may not be easily represented in either continuous- or discrete-valued data. Fuzzy data, which are always represented as linguistic terms, can cope with information containing uncertainty, imprecision and vagueness. Therefore, more complex problems can be modeled when clustering algorithm is able to handle fuzzy input data in addition to continuous- and discrete-valued data. Besides the mentioned data types, we also intend to enhance our clustering algorithm to handle data in a structured and organized manner [Han, Cai, and Cercone 1993]. This includes primitive data, which are stored in data relations and non-primitive data, which appear only in concept hierarchy.

Other than incorporating the ability to handle fuzziness, the determination of suitable or optimal number of clusters will also be investigated. The problem is that it is sometimes quite difficult or impossible to know the number of clusters for causal users especially when domain knowledge about the problem does not exist. Also, given a set of data, it may contain one or more meaningful groupings. Therefore, a clustering algorithm, which is able to determine the number of clusters automatically and discover all possible meaningful groupings, would be very useful for data mining applications.

Finally, to cope with the problem concerned with extremely large databases, the idea of performing data sampling can be employed. Data sampling has been proposed for mining association rules [Zaki *et al.* 1997 and Lin and Dunham 1998]. It can speed up the mining process by reducing I/O costs and the number of records to be considered. The result using sampling in mining association rules shows that sampling can accurately represent the data patterns in the database with high confidence. We therefore consider sampling as one of the feasible solutions to handle very large databases. In addition, how our proposed clustering algorithm can be exploited to make full use of it, parallel nature can be investigated. It is because GA employed in our algorithm can easily be implemented on parallel computer architecture and can be made computationally very efficient.

References

R. Agrawal, S. Ghosh, T. Imielinski, B. Iyer, and A. Swami (1992), "An Interval Classifier for Database Mining Applications," in *Proc. of the 18th VLDB Conf.*, Vancouver, British Columbia, Canada, pp.560-573.

R. Agrawal, T. Imielinski, and A. Swami (1993), "Mining Association Rules between Sets of Items in Large Databases," in *Proceeding of the 1993 ACM SIGMOD International Conference on Management of Data*, Washington D.C., pp. 207-216.

R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, and A.I. Verkamo (1996), "Fast Discovery of Association Rules," in U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy (1996), editors. *Advances in Knowledge Discovery and Data Mining*. MIT Press, pp. 307-328.

R. Agrawal, R. Srikant (1994), "Fast Algorithms for Mining Association Rules," *Proceedings of the 20th International Conference on Very Large Databases*, Santiago, Chile, September.

M.R. Anderberg (1973), *Cluster Analysis for Applications*. Academic Press, New York.

J.R. Anderson and M. Matessa (1991), "An Incremental Bayesian Algorithm for Categorization," in *Concept Formation: Knowledge and Experience in Unsupervised Learning*, edited by D.H. Fisher, M.J. Pazzani and P. Langley, pp.45-70, Morgan Kaufmann Publisher, Inc., San Mateo, California.

Apte (1997), "Data Mining: An Industrial Research Perspective", *IEEE Computational Science & Engineering*, pp.6-9, April-June.

W.H. Au (1998), *DMACK : a system for mining association and classification rules from databases*, M.Phil. Thesis, Dept. of Computing, The Hong Kong Polytechnic University.

D. BarBara, W. DuMouchel, C. Faloutsos, P.J. Haas, J.M. Hellerstein, Y. Ioannidis, H.V. Jagadish, T. Johnson, R. Ng, V. Poosala, K.A. Ross and K.C. Sevcik (1997), "The New Jersey Data Reduction Report", *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*, December.

J.C. Bezdek, S. Boggavarapu, L.O. Hall and A. Bensaid (1994), "Genetic Algorithm Guided Clustering," *IEEE World Congress on Computational Intelligence in Proceedings of the first IEEE Conference on Evolutionary Computation*, pp.34-39, Vol. 1.

G. Biswas, J.B. Weinberg and D.H. Fisher (1998), "ITERATE: A Conceptual Clustering Algorithm for Data Mining," *IEEE Transactions on Systems, Man, and Cybernetics – Part C: Applications and Reviews*, Vol 28, no. 2, May.

P.S. Bradley, U.M. Fayyad (1998), "Refining Initial Points for K-Means Clustering", Proc. Of *the 15th International Conference on Machine Learning ICML'98*, pp.91-99, Morgan Kaufmann, San Francisco.

I. Bhandari, *et al.* (1997). "Advanced Scout: Data Mining and Knowledge Discovery in NBA Data" (summary note). *Data Mining and Knowledge Discovery* **1**(1).

R. Brachman, T. Khabaza, W. Kloesgen, G. Piatesky-Shapiro, and E. Simoudis (1996), "Industrial Applications of Data Mining and Knowledge Discovery", *Communication of the ACM* **39**(11), November.

P. Cabena (1998), *Discovering Data Mining: From Concept to Implementation*, Upper Saddle River, N.J.: Prentice Hall.

K.C.C. Chan and L.L.H. Chung (1999), "Discovering Clusters in Databases Containing Mixed Continuous and Discrete-Valued Attributes," in *Proc. of SPIE AeroSense'99 Data Mining and Knowledge Discovery: Theory, Tools, and Technology*, 5-9 April, pp.22-31.

K.C.C. Chan, A.K.C. Wong and D.K.Y. Chiu (1988), "OBSERVER: a probabilistic learning system for ordered events," *Pattern recognition*. Edited by J. Kittler., pp.507-517, Springer-Verlag, London, United Kingdom.

K.C.C. Chan and A.K.C. Wong (1990), "APACS: a system for automated pattern analysis and classification," *Computational Intelligence*, Vol. 6, pp.119-131.

P. Cheesman and J. Stutz (1995), "Bayesian Classification (AUTOCLASS): Theory and Results", in *Advances in Knowledge Discovery and Data Mining*, U.M. Fayyad, G.P. Shapiro, P. Smythe, and R. Uthurusamy, eds., Ch.6, AAAI/MIT Press.

P. Cheesman, J. Kelly, M. Self, J. Stutz, W. Taylor and D. Freeman (1988), "A Bayesian classification system," *Proceedings of the Fifth International*

Conference on Machine Learning, pp. 54-64, Ann Arbor, MI: Morgan Kaufmann.

M.-S. Chen, J. Han, and P.S. Yu (1996). "Data Mining: An Overview from A Database Perspective," *IEEE Trans. Knowledge and Data Engineering*, vol. 8, no. 6, pp.866-883.

Y. Cheng and K.S. Fu (1985), "Conceptual clustering in knowledge organization," *IEEE Tranactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-7, pp.592-598.

J.Y. Cheung, A.K.C. Wong, and K.C.C. Chan (1995), "Class-Dependent Discretization for Inductive Learning from Continuous and Mixed-Mode Data," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 17, no. 6, pp.1-11.

D.K.Y. Chiu, B. Cheung and A.K.C. Wong (1990), "Information synthesis based on hierarchical maximum entropy discretization," *Journal of Experimental and Theoretical Artificial Intelligence*, Vol. **2**, pp.117-129.

L.L.H Chung and K.C.C Chan, "Discovering Fuzzy Clusters in Using An Evolutionary Approach," submitted.

L.L.H Chung and K.C.C Chan, "Discovering of overlapping clusters in Databases," submitted.

P. Clark, and T. Niblett (1987), "Induction in noisy domains," In Progress in machine learning. *Proceeding of EWSL 87, Second European Working Session on Learning*, Bled, Yugoslavia, Edited by I. Bratko and N. Larvac, pp.11-30.

L. Davis (1991), *Handbook of Genetic Algorithms*. New York: VanNostrand Reinhold.

J.E. Dayhoff (1990), *Neural network architectures : an introduction*, Van Nostrand Reinhold, New York.

K.A. DeJong (1988), "Learning with Genetic Algorithm: An Overview," *Machine Learning*, Vol. 3, pp.121-138.

Diederich and Joachim (1990), *Artificial neural networks : concept learning*, IEEE Computer Society Press, Los Alamitos, California.

A.W.F. Edwards, and L.L. Cavalli-Sforza (1965), "A Method for Cluster Analysis, *Biometric* **21**, pp.362-76.

R. Elmasri and S.B. Navathe (1994), *Fundamentals of Database Systems*, 2nd edition, The Benjamin/Cummings Publishing Company, Inc, CA, NY.

M. Ester, H.-P. Kriegel, and X. Xu (1995), "Discovery in Large Spatial Databases: Focusing Techniques for Efficient Class Identification", *Proceedings of the 4th International Symposium on Large Spatial Databases (SSD'95)*, pp.67-82, Portland, Maine, August, 1995.

M. Ester, H.P. Kriegel, X. Xu (1996), "Knowledge Discovery in Large Spatial Databases: Focusing Techniques for Efficient Class Identification", *Proc. 2nd Internation Conference on Knowledge Discovery and Data Mining*, pp.226-231.

M. Ester, H.-P. Kriegel, J. Sander, X. Xu (1996), "A Density-Based Algorithm for Discovery Clusters in Large Spatial Databases with Noise," *Proceedings of 2nd International Conference on Knowledge Discovery and Data Mining (KDD-96).*

B. S. Everitt (1993), *Cluster Analysis,* 3rd edition, Edward Arnold, New York.

U. Fayyad (1996), "Data Mining and Knowledge Discovery: Making Sense Out of Data," *IEEE Expert*, pp.20-25, Oct.

U. Fayyad (1997), "Data Mining and Knowledge Discovery in Databases: Implications for Scientific Databases", *Proc. 9th International Conference on Scientific and Statistical Database Management*, pp.2-11.

U. Fayyad (1998a), "Taming the Giants and the Monsters: Mining Large Databases for Nuggets of Knowledge," *Database Programming and Design*, Vol. 11, no. 3, March.

U. Fayyad (1998b), "Mining Databases: Towards Algorithms for Knowledge Discovery," *IEEE Bulletin of the Technical Commitee on Data Engineering*, Vol 21, no. 1, March 1998, IEEE Computer Society.

U. Fayyad, D. Haussler, and P. Stolorz (1996a), "Mining Science Data", *Communications of the ACM* **39**(11), November.

U. Fayyad, D. Haussler, and P. Stolorz (1996b), "KDD for Science Data Analysis: Issues and Examples," *Proceedings of Second International Conference on Knowledge Discovery and Data Mining (KDD-96)*, AAAI Press.

U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy (1996), editors. *Advances in Knowledge Discovery and Data Mining*. MIT Press.

U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth (1996a), "Knowledge Discovery and Data Mining: Towards a Unifying Framework," *Proceedings of Second International Conference on Knowledge Discovery and Data Mining (KDD-96)*, AAAI Press.

U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth (1996b), "From Data Mining to Knowledge Discovery in Databases," *AI MAGAZINE*, Fall, pp.37-54.

U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth (1996c), "KDD for Science Data Analysis: Issues and Examples," *Proceedings of Second International Conference on Knowledge Discovery and Data Mining (KDD-96)*, AAAI Press.

U. Fayyad, Cory Reina, and P.S. Bradley (1998), "Initialization of Iterative Refinement Clustering Algorithms", in *the fourth International Conference on Knowledge Discovery and Data Mining*, New York City, August 27-31.

D.H. Fisher, M.J. Pazzani and P. Langley (1991), *Concept formation : knowledge and experience in unsupervised learning*, San Mateo, California, Morgan Kaufmann Publishers.

D.H. Fisher (1987), "Knowledge Acquisition Via Incremental Conceptual Clustering," *Machine Learning*, **2**(2), pp.139-172.

D. Fisher, L. Xu, J.R. Carnes, Y. Reich, S.J. Fenves, J. Chen, R. Shiavi, G. Biswas and J. Weinberg (1993), "Applying AI Clustering to Engineering Tasks", *IEEE Expert*, pp.51-60, December.

E.W. Forgy (1965), "Cluster analysis of multivariate data: Efficiency versus interpretability of classification", *Biometrics* **21**, 3.

R. Forsyth (1990), *PC/BEAGLE User's Guide*, Pathway Research Ltd.

W.J. Frawley, G. Piatetsky-Shapiro, and C.J. Matheus (1991), "Knowledge Discovery in Databases: An Overview," in G. Piatetsky-Shapiro, and W.J. Frawley (Eds.), *Knowledge Discovery in Databases*, AAAI/MIT Press.

V. Ganti, J. Gehrke and R. Ramakrishnan (1999), "CACTUS – Clustering Categorical Data using Summaries," *Proceedings of the fifth ACM SIGKDD*

International Conference on Knowledge discovery and data mining, San Diego, CA USA, Aug 15-18, pp73-83.

J.H. Gennari, P. Langley and D. Fisher (1990), "Model of Incremental Concept Formation," in *Machine Learning – Paradigms and Methods*, edited by J. Carbonell, pp.11-61, MIT, Elsevier, Amsterdam, Netherlands.

M.A. Gluck and J.E. Corter (1985), "Information uncertainty, and the utility of categories," *Proceedings of the Seventh Annual Conference of the Cognitive Science Society*, Irvine, CA: Lawrence Erlbaum Associates, pp.283-287.

D. Gibson, J. Kleinberg and P. Raghavan (2000), "Clustering categorical data: an approach based on dynamical systems," 24th Annual International conference on Very Large Data Baees (VLDB'98), *VLDB Journal* vol. 8, no.3-4, pp. 222-236

D.E. Goldberg (1989), *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley.

L.O. Hall, I.B. Özyurt, and J.C. Bezdek (1999), "Clustering with a Genetically Optimized Approach," *IEEE Transactions on Evolutionary Computation*, Vol. 3, no. 3, July.

J. Han, Y. Cai, and N. Cercone (1993), "Data-Driven Discovery of Quantitative Rules in Relational Databases, " *IEEE Transactions on Knowledge and Data Engineering*, Vol.5, no.1, February.

E. Han, G. Karypis, V. Kumar and B. Mobasher (1998), "Hypergraph Based Clustering in High-Dimensional Data Sets: A Summary of Results", *Bulletin of the IEEE Computer Society Technical committee on Data Engineering*, Vol. 21, no. 1, pp.15-22, IEEE Computer Society, March.

S.J. Hanson and M. Bauer (1989), "Conceptual clustering, categorization, and polymorphy," in *Machine Learning*, **3**:343-372.

S.R. Hedberg (1995), Parallelism speeds data mining, *IEEE Parallel & Distributed Technology*.

Z. Huang (1997a), "Clustering Large Data Sets with Mixed Numeric and Categorical values", *First Pacific-Asia Conference on Knowledge Discovery & Data Mining*, pp.21-34, February.

Z. Huang (1997b), "A Fast Clustering Algorithm to Cluster Very Large Categorical Data Sets in Data Mining," *DMKD*.

Z. Huang (1998), "Extensions to the *k*-Means Algorithm for Clustering Large Data Sets with Categorical Values", *Data Mining and Knowledge Discovery* **2**(3), pp.283-304.

T. Imielinski, and A. Virmani (1995), "DataMine – Interactive Rule Discovery System," in *Proc. of the 1995 ACM SIGMOD Int'l Conf. On Management of Data*, San Jose, CA, pp.472.

A.K. Jain and R.C. Dubes (1988), *Algorithms for Clustering Data*. Prentice Hall.

T. Jiang and S. De Ma (1996), "Cluster Analysis Using Genetic Algorithms," in *Proceedings of International Conference on Signal Processing ICSP'96*, pp.1277-1279, Vol.2.

L. Kaufman and P.J. Rousseeuw (1990), *Finding Groups in Data: an Introduction to Cluster Analysis*, John Wiley and Sons.

C.H.M. van Kemenade (1996), "Cluster Evolution Strategies," in *Proceedings of IEEE International Conference on Evolutionary Computation 1996*, pp.637-642.

R.L Kennedy, Y. Lee, B.V. Roy, C.D. Reed, and R.P. Lippman (1997), *Solving Data Mining Problems Through Pattern Recognition*, The Data Warehousig Institute Series, Prentice Hall PTR, Upper Saddle River, New Jersey.

R. Kohavi (1996), "Scaling Up the Accuracy of Naïve-Bayes Classifiers: A Decision-Tree Hybrid," in *Proc. of the 2nd International Conference on Knowledge Discovery and Data Mining*, pp.202-207.

T. Kohonen (1984), *Self-organization and associative memory*, Springer-Verlag, Berlin , New York.

T. Kohonen (1997), *Self-organizing maps*, Springer, Berlin , New York.

K. Krishna and M.N. Murty (1999), "Genetic K-Means Algorithm," *IEEE Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics*, Vol. 29, no.3, June.

R. Krovi (1992), "Genetic Algorithms for Clustering: A Preliminary Investigation," *Proceedings of the Twenty-Fifth Hawaii International Conference on System Sciences*, pp.540-544, Vol.4.

L.I. Kuncheva and J.C. Bezdek (1998), "Nearest Prototype Classification: Clustering, Genetic Algorithm, or Random Search?," *IEEE Transactions on Systems, Man, and Cybernetics – Part C: Applications and Reviews*, Vol. 28, no. 1, February.

M. Lebowitz (1987), "Experiments with incremental concept formation: UNIMEM," in *Machine Learning*, **2**:103-138.

C. Li and G. Biswas (1997), "Unsupervised Clustering with Mixed Numeric and Nominal Data – A New similarity Based Agglomerative System", *First Pacific-Asia Conference on Knowledge Discovery & Data Mining*, pp.35-48, February.

J.L. Lin and M.H. Dunham (1998), "Mining association rules: anti-skew algorithms," *Proceedings of the 14th International Conference on Data Engineering*, pp.486-493.

H. Lu, R. Setiono, and H. Liu (1995), "NeuroRule: A Connectionist Approach to Data Mining," in *Proc. of the 21st VLDB Conf.*, Zurich, Switzerland, pp.478-489.

H. Lu, R. Setiono, and H. Liu (1996), "Effective Data Mining Using Neural Networks," *IEEE Transactions on Knowledge and Data Engineering*, Vol. 8, no. 6, pp.957-961.

J.B. MacQueen (1967), "Some methods for classification and analysis of multivariate observations," *Proc. 5th Berkeley Symp. on Mathematical Statistics and Probability*, 1, pp.281-297.

D.J. Marshall (1999), "Data Mining Using Genetic Algorithms," in *Industrial Applications of Genetic Algorithms*, edited by C.L. Karr and L.M. Freeman, CRC Press, NY, pp.159-194.

C.J. Matheus, P.K. Chan, and G. Piatetsky-Shapiro (1993), "System for Knowledge Discovery in Databases," *IEEE Transactions on Knowledge and Data Engineering*, Vol. 5, no. 6, pp.903-913.

G. Matthews and J. Hearne (1991), "Clustering Without a Metric", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol 13, No. 2, February.

Z. Michalewicz (1996), *Genetic Algorithms + Data Structures = Evolution Programs*, Third, Revised and Extended Edition, Springer.

D. Michie, D.J. Spiegelhalter, and C.C. Taylor (Eds.) (1994). *Machine Learning, Neural, and Statistical Classification*, Ellis Horwood.

R.S. Michalski (1980), "Knowledge acquisition through conceptual clustering: A theoretical framework and algorithm for partitioning data into conjunctive concepts," *International Journal of Policy Analysis and Information System*, **4**, pp.210-243.

P. Michaud (1997), "Clustering techniques," *Future Generation Computer System* **13**(2-3), pp.135-147, Elserier, Netherlands, November.

F. Murtagh (1985), *Multidimensional Clustering Algorithms*, Physica-Verlag, Vienna.

R.T. Ng, J. Han (1994), "Efficient and Effective Clustering Methods for Spatial Data Mining," *Proceedings of the 20th International Conference on Very Large Databases*, Santiago, Chile, September, pp.144-155.

M.O. Noordewier, G.G. Towell, and J.W. Shavlik (1991), "Training Knowledge-Based Neural Networks to Recognize Genes in DNA Sequences," in R.T. Lippmann, J.E. Moody, and D.S. Touretzky (Eds.), *Advances in Neural Informaiton Processing Systems*, Vol. 3, Morgan Kaufmann.

Y.H. Pao, and C.H. Hu, Processing of pattern-based information: Part I: inductive inference methods suitable for use in pattern recognition and artificial intelligence. *In advances in information system sciences*, Vol.9, Edited by J.T. Tou, Plenum Press, New York, NY, 1984.

Y.H. Pao, W.L. Schultz, M. Kiley, J.L. Altman, and A.J.L. Schneider, Implementation of human judgement and experience in computer-aided interpretation of medical images. *Proceedings of the 3rd International Joint Conference on Pattern Recognition*, pp.228-232, 1976.

G. Piatetsky-Shapiro, and W.J. Frawley (1991), Editors, *Knowledge Discovery in Databases*, AAAI/MIT Press.

J.R. Quinlan (1987), "Simplifying Decision Trees," *International Journal of Machine Studies*, col. 27, pp.221-234.

J.R. Quinlan (1993), *C4.5: Programs for Machine Learning*, Morgan Kaufmann.

H. Ralambondrainy (1995), "A conceptual version of the K-means algorithm," *Pattern Recognition Letter*, no.16, pp.1147-1157.

G.D. Ramkumar and A. Swami (1998), "Clustering Data Without Distance Functions", *Bulletin of the IEEE Computer Society Technical committee on Data Engineering*, Vol. 21, no. 1, pp.9-14, IEEE Computer Society, March.

Shapiro and C. Stuart (1992), *Encyclopedia of artificial intelligence*, 2nd edition, A Wiley-Interscience publication, New York.

A. Silberschatz, M. Stonebraker, and J. Ullman (1991), "Database Systems: Achievement and Opportunities," *Communications of the ACM*, Vol. 34, no.10, pp.94-109.

A. Silberschatz, M. Stonebraker, and J. Ullman (1996), "Database Research: Achievement and Opportunities Into the 21th Century," *SIGMOD Record*, Vol.25, no.1.

J.W. Smith, J.E. Everhart, W.C. Dickson, W.C. Knowler, and R.S. Johannes (1988), "Using the ADAP Learning Algorithm to Forecast the Onset of Diabetes Mellitus," in *Proc. of the Symp. On Computer Appliations and Medical Cares*, pp.261-265.

W. Song, M. Feng, S. Wei, and X. Shaowei (1997), "The hyperellipsoidal Clustering Using Genetic Algorithm," *IEEE International Conference on Intelligence Processing Systems*, Oct 28-31, Beijing, China, pp.592- 596.

H. Spath (1980), *Cluster Analysis Algorithms for Data Reduction and Classification of Objects*, Ellis Horword, Chichester.

R.E. Stepp (1987), "Concepts in conceptual clustering," *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, Milan, Italy: Morgan Kaufmann, pp.211-213.

M.C. Su and H.T. Chang (1998), "Genetic-Algorithms-Based Approach to Self-Organizing Feature Map and its Application in Cluster Analysis," *IEEE World Congress on Computational Intelligence. The 1998 IEEE International Joint Conference on Neural Network*, Vol. 1, pp.735-740.

G. Syswerda (1989), "Uniform crossover in genetic algorithms," in *Proc. 3rd Int. Conf. Genetic Algorithms*. p.2, San Mateo, CA: Morgan Kaufmann.

L.Y. Tseng and S.B. Yang (1997), "Genetic Algorithms for Clustering, Feature Selection and Classification," *International Conference on Neural Networks*, Vol.3, pp.1612-1616.

J.D. Ullman (1989), *Principle of Database and Knowledge-Base System*, Vol. 1 & 2, Computer Science Press.

J.H. Ward (1963), *Hierarchical grouping to optimize an objective function*, J.Amer. Statist. Assoc., pp.236-244.

D. Whitley and J. Kauth (1988), "Genitor: A different genetic algorithm," in *Proc. Rocky Mountain Conf. Artificial Intelligence*, Denver, CO, p.118.

A.K.C. Wong, D.C.C. Wang (1979), "DECA: A Discrete-Valued Data Clustering Algorithm," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-1, no.4, October.

A.K.C. Wong and D.K.Y. Chiu (1987), "Synthesizing statistical knowledge from incomplete mixed-mode data," *IEEE Transaction on Pattern Analysis and Machine Intelligence*, Vol. 9, no. 6, pp.796-806, November.

A. Wu, K.Y. Wu, R.M.M. Chen, and Y. Shen (1998), "Parallel Optimal Statistical Design Method With Response Surface Modelling Using Genetic Algorithm," *IEE Proceedings – Circuits Devices Systems*, Vol.145, no.1, Feb.1998, pp.7-12.

L.A. Zadeh (1965), "Fuzzy sets," *Information and Control*, Vol.8, pp.338-353.

M.J. Zaki, S. Parthasarathy, Wei Li, and M.Ogihara (1997), "Evaluation of sampling for data mining of association rules," *Proceedings of the Seventh International Workshop on Research Issues in Data Engineering*, pp.42-50.

T. Zhang, R. Ramakrishnan, and M. Livny (1996), "BIRCH: An efficient data clustering method for very large databases," In *Proceedings of the ACM-SIGMOD Conference on Management of data, Montreal*, Canada.