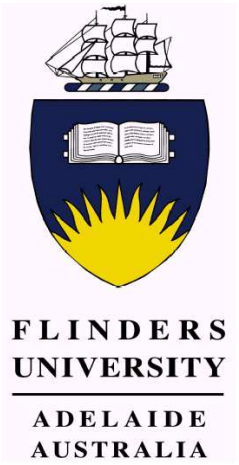


Improved Clustering and Soft Computing Algorithms

by

Shu-Chuan Chu, *B.Sc.(Hons)*
School of Informatics and Engineering,
Faculty of Science and Engineering

January 1, 2004

A thesis presented to the
Flinders University of South Australia
in total fulfillment of the requirements for the degree of
Doctor of Philosophy

Adelaide, South Australia, 2004

Contents

List of Figures

List of Tables

Abstract

Clustering algorithms have been widely applied and different approaches have been developed for different domains of application. This thesis investigates efficient and effective clustering and soft computing algorithms. In the investigation of k-medoids algorithms, several improved algorithms are proposed, such as the *Clustering Large Applications Based on Simulated Annealing* ($CLASA$) algorithm, the *Multi-Centroids with Multi-Runs Sampling Scheme* ($MCMRS$) and *Incremental Multi-Centroid, Multi-Run Sampling Scheme* ($IMCMRS$) algorithms. The *Partial Distance Search* (PDS), *Triangular Inequality Elimination* (TIE) and *Previous Medoid Index* are also presented to improve the clustering speed of k-medoids based algorithms. In addition, a new memory utilization scheme is derived and applied to efficient k-medoids algorithms.

In the investigation of centroid-based clustering algorithms, the tabu search with simulated annealing algorithm is proposed and applied to codebook design for vector quantization. Genetic clustering is also presented for mean-residual vector quantization. Several theorems based on Hadamard Transform for nearest neighbour search are presented and applied to efficient cluster (codeword) search for vector quantization. A label bisecting clustering algorithm is proposed and applied to create a robust watermarking technique.

Parallel particle swarm optimization based on three communication strategies are proposed to solve the problems in which the relationship between parameters are either independent, loosely correlated, strongly correlated or unknown. Seven communication strategies for *Ant Colony Systems* (ACS) are proposed to improve the ACS for the traveller salesman problem and the *Constrained Ant Colony Optimization* ($CACO$) based on the quadratic metric, sum of k nearest neighbour distance, constrained addition of pheromone and a shrinking range strategy is also proposed and demonstrated to be better than the *Ant Colony Optimization with Different Favor* ($ACODF$).

Certification

I certify that this thesis does not incorporate without acknowledgement any material previously submitted for a degree or diploma in any university; and that to the best of my knowledge and belief it does not contain any material previously published or written by another person except where due reference is made in the text.

As requested under Clause 14 of Appendix D of the *Flinders University Research Higher Degree Student Information Manual* I hereby agree to waive the conditions referred to in Clause 13(b) and (c), and thus

- Flinders University may lend this thesis to other institutions or individuals for the purpose of scholarly research;
- Flinders University may reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

Signed Dated

Shu-Chuan Chu

Acknowledgements

I would like to first thank my supervisor, Professor John F. Roddick, for his constant guidance, patience and encouragement.

I would like to thank the principal of National Kaohsiung University of Applied Sciences, Professor Jen-Yi Lin, and the former principal, Professor Kuang-Chih Huang without whose high level of support and help, I cannot contentedly continue to finish this work in my academic career. Further, I would like to extend my thanks to other colleagues and friends who have provided help and advice during my PhD study. Without particular order, thanks to: Professor Wen-Kuei Cheng, Professor Bin-Yi Liao, Chin-Shiuh Shieh and Chin-Yen Chang.

Finally, I wish to dedicate this thesis to my husband, my mother and my parents in law who have guided me all through my life. I would like to thank them for all the love, suggestions and encouragements.

Shu-Chuan Chu
January 2004
Adelaide.

Chapter 1

Introduction

Clustering in data mining is used to group similar objects based on their distance, connectivity, relative density, or some specific set of characteristics. The clustering methods have been mentioned in many contexts (Jain & Dubes 1988, Kaufman & Rousseeuw 1990, Hartigan 1975) and the applications of clustering include numerous areas, including pattern recognition (Anderberg 1973), character recognition (Babu & Murty 2001), image processing (Jain & Flynn 1996), computer vision (Jolion, Meer & Bataouche 1991), information retrieval (Rasmussen 1992), web search (Krishnapuram, Joshi & Yi 1999, Mobasher, Cooley & Srivastava 1999, Zamir, Etzion, Mandani & Karp 1997), Engineering (Fisher, Xu, Carnes, Reich, Fenves, Chen, Shiavi, Biswas & Weinberg 1993), Geographic Information Systems (Estivill-Castro & Lee 2000*a*, Estivill-Castro & Lee 2000*b*, Han, Kamber & Tung 2001, Ng 1996, Sander, Ester, Kriegel & Xu 1998), Image Compression (Gersho & Gray 1992), OLAP (Markl, Ramsak & Bayer 1999), Protein Sequencing (Enright & Ouzounis 2000), Psychiatry (Lecompte, Kaufman & Rousseeuw 1986) and Vector quantization (Pan, McInnes & Jack 1996*c*). Various clustering algorithms have been designed to fit various requirements and constraints of application. Clustering techniques can also be applied to recognize different geometrical structures. Su and Chou (Su & Chou 2001) proposed a modified k-means algorithm based on point symmetry distance measure to cluster

the ellipsoidal shells and detect human faces.

A number of clustering scenarios have caused problems in the past. Many have been solved but others still cause some problems, see Figure 1.1.

- **Non-spherical clusters**. Clustering algorithms such as k-means and k-medoids naturally form spherical clusters. This means that some irregular shapes, such as those shown in Figure 1.1(a), would be erroneously clustered together. This problem has largely been solved through grid-based, hierarchical and density based clustering techniques such as, for example, $CLIQUE$ (Agrawal, Gehrke, Gunopulos & Raghavan 1998), $CURE$ (Guha, Rastogi & Shim 1998) and $DBSCAN$ (Ester, Kriegel, Sander & Xu 1996) respectively.

- **Shaped clusters**. In some applications, the shape of the eventual clusters are significant and many algorithms fail to account for this. For example, it is likely that many algorithms would consider Figure 1.1(c) to consist of two rather than four clusters.

- **Clusters with outliers**. While the k-medoids method is considered more robust to (the influence of) outliers, some forms of outlier can still cause problems or unduely influence the results of clustering. Moreover, in some work it is the outliers that are the focus of interest.

- **Bridges**. Figures 1.1(g-i) shows three scenarios in which the bridge between two or three obvious clusters need to be handled carefully. In many cases the bridge itself is required to be identified as a cluster separately.

- **Constraints**. Although a real-world requirement, the existence of barriers or other forms of constraint has only relatively recently been considered (Zaïane & Lee 2002).

- **Non-parametric clustering**. The need to specify, in advance, parameters such as the number of clusters to form can be a problem. The acceptable

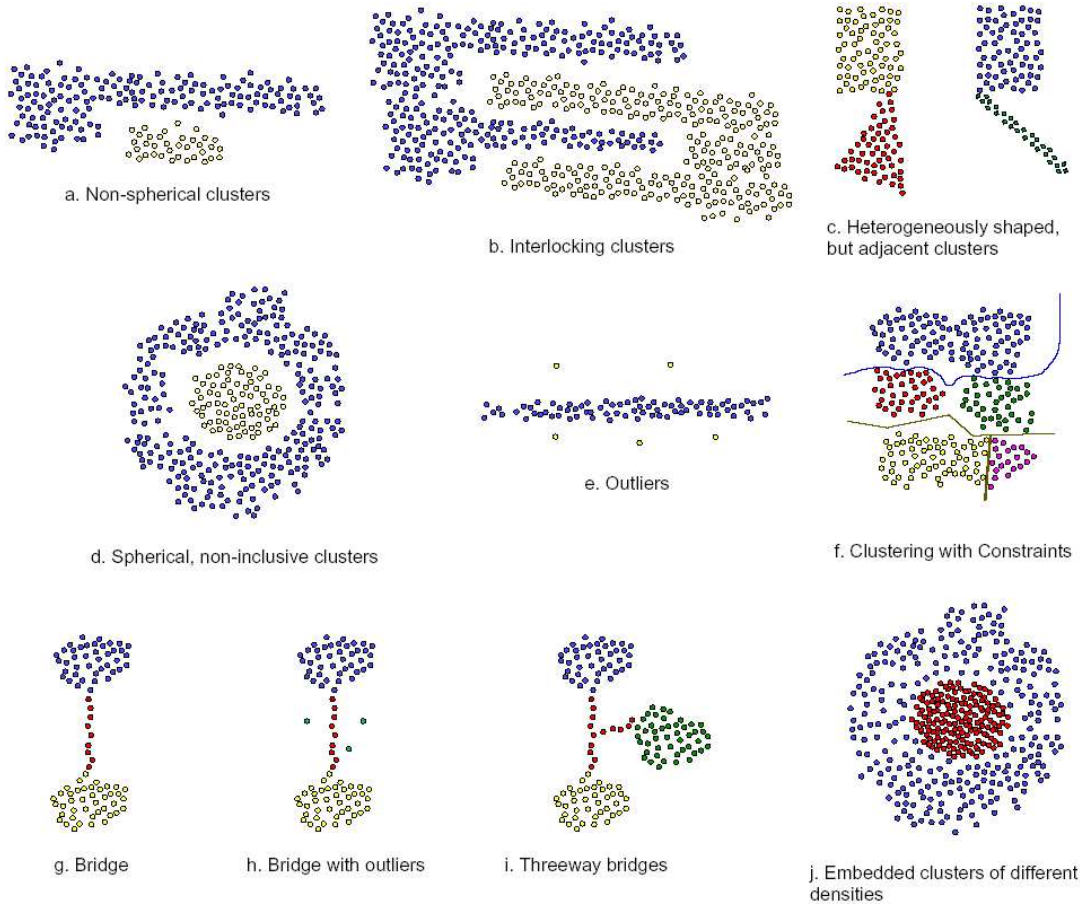


Figure 1.1. Problem datasets for some clustering algorithms

criterion, for example, for a neighbour, the required cluster density, etc. is dealt with by some routines but some issues remain a problem.

- **Clusters of varying density**. Some routines have problems finding clusters of varying density at the same time, such as those indicated in Figures 1.1(d) and 1.1(j).

Texture analysis is of high importance for the visual-scene analysis because an entire image can be interpreted as a composition of different textures (Pan & Wang 1999). How to segment the texture is an important issue for research. The texture segmentation can be formulated as a combinatorial optimization problem which can be solved using clustering technique. The tabu search approach has been applied to segment the texture (Pan, Wang, Fang & Chen 1998). A modified

k-means algorithm based on spatial and wavelet domains is also applied to texture segmentation (Ng & Bouzerdoun 2001).

In business, a market basket database contains transaction per customer and each transaction contains a set of items purchased by the customer, marketer can utilize these transaction data to cluster the customers such that customers with similar buying items are in the same cluster. The clusters can then be used to classify into the different customer groups, and these characterizations can be used in targeted marketing and advertising such that specific products are directed towards specific customer groups. The characterizations can also be used to predict buying patterns of new customers (Han, Karypis, V. & Mobasher 1997, Guha, Rastogi & Shim 1999).

During the last decade, with the prevalence of Internet access and usage, digital media, including images, audio, and video sequences, are easily acquired in our daily life. Owing to the digital nature of such media, the unlimited copying and easy distribution have made copyright protection an important topic. The clustering techniques can be applied to embed the watermark in a suitable position of the original image such that the watermarked image can be robust to some attacks (Lu, Pan & Sun 2000c, Lu & Sun 2000).

Soft computing is an emerging collection of methodologies to exploit tolerance for uncertainly, imprecision and partial truth to achieve useable robustness, tractability, low total cost and approximate solutions. It is particularly efficient and effective for NP-hard problems. Recently, many differently challenges posed by data mining have been solved by various soft computing methodologies. At this juncture, the main components of soft computing involve fuzzy theory, artificial neural networks, genetic algorithms, simulating annealing, tabu search approach, swarm intelligence systems (such as particle swarm optimization, ant systems and ant colony systems) and other approaches related to cognitive modelling. Each of them contributes a revealable methodology which only in a cooperative rather than competitive manner for persuading problems in its field. It is ex-

pected that soft computing techniques will finally become as general and popular as traditional methods of computer science.

Because there are many proposed clustering techniques to solve difficult problems in various areas, clustering has possessed of an influential position. In this thesis, we focus on developing some novel efficient and effective soft computing techniques and applying various soft-computing tools and their hybridizations for improving clustering algorithms.

1.1 Related Existing Clustering Algorithms

Clustering is a dynamic field of research in data mining. Lots of clustering algorithms have been proposed and discussed in many contexts and disciplines. These can be classified into *inter alia* partitioning methods, hierarchical methods, density-based methods, grid-based methods and model-based methods as described as follows (Han & Kamber 2001):

Partitioning Methods:

Given a database of n objects, a partitioning method creates k partitions of the objects, where each partition represents a cluster. Each cluster contains at least one object and each object must belong to exact one group. Both k-means (MacQueen 1967) (which adopts as the representative point the weighted mean of the cluster) and k-medoids (Kaufman & Rousseeuw 1990) (which adopts as the representative point the most central object in the cluster) algorithms are typical partitioning methods. Centroid-based clustering technique such as k-means (or LBG, GLA) (MacQueen 1967) has been applied to generate codebook from a large spatial database. The codebook can be used for image coding, speech recognition and data compression. Vector quantization is one of the popular data compression methods. One of the core techniques for vector quantization is to generate useful codebook

which mainly depends on the clustering techniques. One of the evaluation criteria in vector quantization aims at generating clusters (or codebook) from a large spatial database so that the average distortion between the input vector and the reconstructed vector can be reduced. PAM was the first method proposed for k-medoids algorithm and had been improved by $CLARA$ (Kaufman & Rousseeuw 1990). Both PAM and $CLARA$ are reported to be inefficient method based on the computational complexity. $CLARANS$ was designed for cluster analysis (Ng & Han 1994). Experimental results demonstrate that $CLARANS$ was superior to these two algorithms. Based on the $CLARANS$, the spatial domain and non-spatial domain clustering algorithms were developed. $CLARANS$ has been applied to find the clusters of Vancouver expensive housing units successfully.

Hierarchical Methods:

A hierarchical method constructs a hierarchical decomposition of the given set of objects. This method gives rise to *dendrogram* in which the patterns are formed a nested sequence of partitions. Hierarchical procedures can be either agglomerative or divisive. An agglomerative clustering approach is a process in which each pattern is placed in its own cluster and these atomic clusters are gradually merged into larger and larger clusters until the desired objective is attained. A divisive clustering approach reverses the process of agglomerative clustering approach by starting with all patterns in one cluster and subdividing into several smaller clusters. *BIRCH* (Zhang, Ramakrishnan & Livny 1996), *CURE* (Guha et al. 1998), *ROCK* (Guha et al. 1999) and *CHAMELEON* (Karypis, Han & Kumar 1999) are all hierarchical methods. BIRCH is the first clustering algorithm proposed in the database area to handle noise effectively. The basic idea of *BIRCH* is to integrate hierarchical agglomeration and iterative relocation by first using a hierarchical agglomerative algorithm and then refining the result using iterative relocation. *CURE* partitions a random sample firstly, and each

partition is partially clustered. The partial clusters are then clustered in a second pass to yield the desired clusters. *CHAMELEON* generates the k-nearest neighbor graph from the objects, then partition and merge the graph based on the criteria of inter-connectivity and closeness to get final clusters.

Density-Based Methods:

Most partitioning methods cluster objects based on the distance between objects, such that only spherical-shaped and/or elliptical-shaped clusters can be found. It is hard to discover clusters of arbitrary shapes. DBSCAN (Ester et al. 1996) is a clustering method based on the density threshold of clusters. The object belong to the same cluster if the distance between this object and any one object in the cluster is smaller than a distance threshold, then this object is set to the same cluster. If the number of objects in the cluster is smaller than a density threshold, then all the objects in this cluster are recognized as the noise (outliers). Clusters of arbitrary shape can be found in this approach and is well appropriate for spatial databases. The other two related density based clustering algorithms are OPTICS (Ordering Points To Identify the Clustering Structure) (Ankerst, Breunig, Kriegel & Sander 1999) and DENCLUE (Clustering Based on Density Distribution Functions) (Hinneburg & Keim 1998). OPTICS generates an augmented ordering of the database to represent its density-based clustering structure. DENCLUE is a distribution-based clustering algorithm according to some influence function to generate clusters.

Grid-based methods:

Grid-based methods quantize the object space into a finite number of grids. The benefit of this method is the efficient processing time. Normally the processing time is independent of the number of objects and dependent only on the number of grids in each dimension in the quantized space.

STING (Wang, Yang & Muntz 1997) is a grid-based method while CLIQUE (Agrawal et al. 1998) and WaveCluster (Sheikholeslami, Chatterjee & Zhang 1998) are clustering algorithm based both on grid-based and density-based methods.

Model-based methods:

Model-based clustering methods attempt to optimize the clusters of the objects based on some mathematical models. Statistical approach and neural network approach are two major approaches for model-based methods. $COBWEB$ is a popular and simple method of incremental conceptual clustering based on the model of statistical approach (Fisher 1987). SOM (self-organizing feature map) is a typical clustering algorithm based on the model of neural network approach (Kohonen 1982).

1.2 Soft Computing

The realm of soft computing encompasses fuzzy logic and other meta-heuristics such as genetic algorithms, neural networks, simulated annealing, particle swarm optimization, ant colony systems and parts of learning theory as shown in Figure 1.2 (Chen 2002). It is expected that soft computing techniques have received increasing attention in recent years for their interesting characteristics and their success in solving problems in a number of fields.

1.2.1 Fuzzy Theorem

Along the trace of development of computing machinery, there was a long history of research trying to endow cybernetics with intelligence. According to the adopted representation scheme, there are two main streams of approaches: symbolic systems and sub-symbolic systems. In symbolic systems, symbols are given

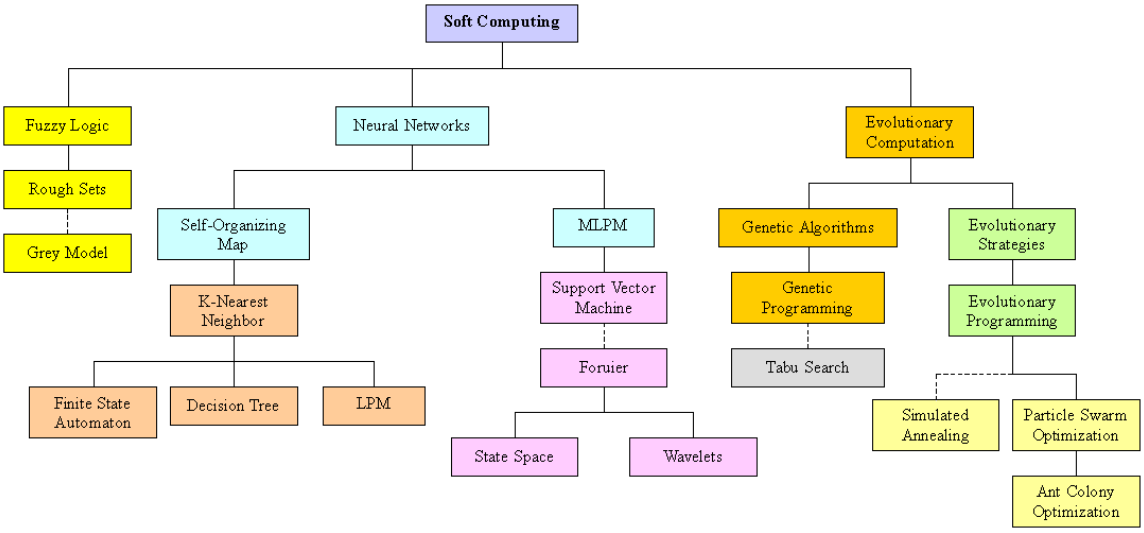


Figure 1.2. The hybridization of soft computing

with semantic meanings to represent concrete or abstract entities and the relationships among them, such as predicate calculus and expert systems. Neural networks and classifier systems are examples of the second category, in which knowledge is distributed to the weight or strength of a vast number of simple units. Formal logic does contribute to the discipline of modern science and engineering, but it did not provide a total solution to the problem faced by the community of artificial intelligence. The obstacle may be, in part, due to that a great portion of real world is inexact rather than precise. Therefore, formal logic alone is insufficient to model the mental activity of human beings. Fuzzy systems, pioneered by Zadeh (Zadeh 1965), stand at the gray zone between symbolic and sub–symbolic systems. Rooted in fuzzy theory, fuzzy systems are trying to cope with the inexact nature of real world and to model the mental activity of human beings.

Fuzzy systems can been distinguished for the following characteristics (Dubois & Prade 1980, Mamdani & Gaines 1981):

1. Rule-based: Benefit of symbolic system was preserved by coding expertise into fuzzy rules described by linguistic variables.

2. Robust: The representation scheme and inference process of fuzzy systems are inherently immune to environmental noise, change in system parameters, and damage of rule base.

3. Easy-prototyping: Workable prototype can be easily constructed, and then tuned according to actual operation data.

4. Highly representative: The fuzzy rules are perceivable to human being. This characteristic is crucial if a fuzzy system can learn new rules or adapt old rules by itself.

5. Both symbolic and numerical: Rules are stored symbolically and inferred numerically making fuzzy systems able to response to wide range of input by only a few rules.

After decades of research effort, fuzzy systems gain from various prospects. Not only the idea was settled down to mathematical foundations (Kosko 1992) but also dedicated hardware was developed (Togai & Watanabe 1986, Yamakawa 1988). Applications in different fields and commercialized product had revealed the potential of fuzzy systems. Fuzzy controller is a successful application of fuzzy theory, especially for ill-structured control problems.

1.2.2 Artificial Neural Networks

Neural networks may be one of the future directions of computing. An Artificial Neural Network (*ANN*) is an information-processing paradigm that is inspired by the way biological nervous systems, such as the brain, process information. The key element of this paradigm is the novel structure of the information processing system. It is composed of a large number of highly interconnected processing elements, termed "neurons" working in unison to solve specific problems.

ANNs are a form of multiprocessor computer system, with simple processing elements, a high degree of interconnection, simple scalar messages, and adaptive

interaction between elements. Like people to learn by example, an *ANN* is configured for specific applications including image processing (Lo, Huai & Freedman 2003, Robinson & Kecman 2003), pattern recognition (Liu & Wechsler 2003), speech processing (Xiang & Berger 2003), communication systems (Palicot & Roland 2003), control engineering (Lavretsky, Hovakimyan & Calise 2003, Park, Harley & Venayagamoorthy 2003), and other areas in engineering, through a learning process. Learning in biological systems involves adjustments to the synaptic connections that exist between the neurons. This is true of *ANNs* as well. *ANNs* are important approaches in the area of soft computation.

1.2.3 Simulated Annealing

The original idea of *Simulated Annealing* (*SA*) was proposed by Kirkpatrick and his colleagues (Kirkpatrick, Gelatt & Vecchi 1983). It is a random search method presented for optimization of NP-hard problems. The idea is analogous to the process of metal annealing. The temperature is introduced to the optimization process so as to control the probability to jump out of the current position. The temperature value may be reduced proportionally to the increase of the number of iterations. In the initial iterations, the temperature is large so as to jumping out of the current position easier. As the iteration number is increased, it tends to be converged to the current position with a small probability to leave the present position. *SA* is capable of escaping from the local optimum due to the use of stochastic random number. *SA* has been successfully applied to power system engineering (Gallego, Alves, Monticelli & Romero 1997), cell placement in *VLSI* design (Kurbel, Schneider & Singh 1998), various codes construction (Gamal, Hemachandra, Shperling & Wei 1987), and the optimization of wiring problem (Vecchi & Kirkpatrick 1983). Simulated annealing is a key approach in the members of soft computation family.

1.2.4 Tabu Search Approach

The Tabu search approach (Skorin-Kapov 1990, Pan & Chu 1996, Glover & Laguna 1997, Chu & Fang 1999, Lu, Pan & Sun 2000*b*, Pan, Lu, Shieh & Sun 2000, Al-Sultan 1995, Franti, Kivijarvi & Nevalainen 1998) is a higher-level method for solving combinatorial optimization problems. It is a metaheuristic developed by Glover (Glover 1977, Glover 1986) and designed to optimize the problem by performing a sequence of moves that lead the procedure from one test solution to another. The idea of tabu searching is to forbid some search directions at a present iteration in order to avoid cycling, so as to jump off local optima. Each move is selected randomly from a set of currently available alternatives. The new test solutions are generated by performing the moves from the current best solution and the current best solution is the test solution which is not a tabu solution or it is a tabu solution but it satisfies the aspiration criterion. The tabu solution is the solution in which the elements of the solution are partially or completely recorded in the tabu list memory. Tabu list memory is used to partially or completely record the elements of move from the current best solution to its selected neighbour. It is called *aspiration* if the test solution is in the tabu condition but it is the best solution for all iterations up to now.

1.2.5 Genetic Algorithms

Based on long-term observation, Darwin asserted his theory of natural evolution. In the natural world, creatures compete with each other for limited resources. Those individuals that survive in the competition have the opportunity to reproduce and generate descendants. In so doing, any exchange of genes may result in superior or inferior descendants with the process of natural selection eventually filtering out inferior individuals and retain those adapted best to their environment.

Inspired by Darwin's theory of evolution, Holland (Holland 1975, Goldberg

1989) introduced the genetic algorithm as a powerful computational model for optimization. Genetic algorithms work on a population of potential solutions, in the form of chromosomes, and try to locate a best solution through the process of artificial evolution, which consist of repeated artificial genetic operations, namely evaluation, selection, crossover and mutation.

Although the operation of genetic algorithms is quite simple, it does have some important characteristics providing robustness:

- They search from a population of points rather than a single point.
- The use the object function directly, not their derivative.
- They use probabilistic transition rules, rather than deterministic ones, to guide the search toward promising regions.

In effect, genetic algorithms maintain a population of candidate solutions and conduct stochastic searches via information selection and exchange. It is well recognized that, with genetic algorithms, near-optimal solutions can be obtained within justified computation cost. However, it is difficult for genetic algorithms to pin point the global optimum. In practice, a hybrid approach is recommended by incorporating gradient-based or local greedy optimization techniques. In such integration, genetic algorithms act as course-grain optimizers and gradient-based method as fine-grain ones.

The power of genetic algorithms originates from the chromosome coding and associated genetic operators. It is worth paying attention to these issues so that genetic algorithms can explore the search space more efficiently. The selection factor controls the discrimination between superior and inferior chromosomes. In some applications, more sophisticated reshaping of the fitness landscape may be required. Other selection schemes (Whitley 1993), such as rank-based selection, or tournament selection are possible alternatives for the controlling of discrimination.

Numerous variants with different application profiles have been developed following the standard genetic algorithm. Island-model genetic algorithms, or parallel genetic algorithms (Abramson & Abela 1991), attempt to maintain genetic diversity by splitting a population into several sub-populations, each of them evolves independently and occasionally exchanges information with each other. Multiple-objective genetic algorithms (Gao & Yao 2000, Fonseca & Fleming 1993, Fonseca & Fleming 1998) attempt to locate all near-optimal solutions by careful controlling the number of copies of superior chromosomes such that the population will not be dominated by the single best chromosome (Sareni & Krahenbuhl 1998). Co-evolutionary systems (Handa & Katai 2002, Bull 2001) have two or more independently evolved populations. The object function for each population is not static, but a dynamic function depends on the current states of other populations. This architecture vividly models interaction systems, such as prey and predator, virus and immune system.

1.2.6 Particle Swarm Optimization

Some social systems of natural species, such as flocks of birds and schools of fish, possess interesting collective behavior. In these systems, globally sophisticated behavior emerges from local, indirect communication amongst simple agents with only limited capabilities.

In an attempt to simulate this flocking behavior by computers, Kennedy and Eberthart (1995) realized that an optimization problem can be formulated as that of a flock of birds flying across an area seeking a location with abundant food. This observation, together with some abstraction and modification techniques, led to the development of a novel optimization technique – particle swarm optimization.

Particle swarm optimization optimizes an object function by conducting a population-based search. The population consists of potential solutions, called particles, which are a metaphor of birds in flocks. These particles are randomly

initialized and freely fly across the multi-dimensional search space. During flight, each particle updates its velocity and position based on the best experience of its own and the entire population. The updating policy will drive the particle swarm to move toward region with higher object value, and eventually all particles will gather around the point with highest object value.

Numerous variants had been introduced since the first particle swarm optimization. A discrete binary version of the particle swarm optimization algorithm was proposed by Kennedy and Eberhart (1997). Shi and Eberhart (2001) applied fuzzy theory to particle swarm optimization algorithm, and successfully incorporated the concept of co-evolution in solving min-max problems (Shi & Krohling 2002).

1.2.7 Ant Systems and Ant Colony Systems

Inspired by the food-seeking behavior of real ants, Ant Systems, attributable to Dorigo *et al.* (Dorigo, Maniezzo & Colorni 1996), has demonstrated itself to be an efficient, effective tool for combinatorial optimization problems. In simplistic terms, in nature, a real ant wandering in its surrounding environment will leave a biological trace - *pheromone* - on its route. A more ants take the same route the level of this pheromone will increase. The intensity of pheromone at any point will bias the path-taking decisions of subsequent ants. After a while, the shorter paths will tend to possess higher pheromone concentration and therefore encourage subsequent ants to follow them. As a result, an initially irregular path from nest to food will eventually focus to form the shortest paths. A more promising method was also developed and referred to as the algorithm of *Ant Colony System* (*ACS*) (Dorigo & Gambardella 1997). With appropriate abstractions and modifications, these natural observations have led to a successful computational model for combinatorial optimization. The Ant System and Ant Colony System have been applied successfully in many applications such as the quadratic

assignment problem (Maniezzo & Colorni 1999), data mining (Parpinelli, Lopes & Freitas 2002), space-planning (Bland 1999), job-shop scheduling and graph coloring (Dorigo, Caro & Gambardella 1999).

In some respects, the ant system has implemented the idea of emergent computation – a global solution emerges as distributed agents performing local transactions, which is the working paradigm of real ants. The success of ant systems in combinatorial optimization makes it a promising tool for dealing with a large set of problems in the NP-complete class (Papadimitriou & Steiglitz 1982). In addition, the work of Wang and Wu (Wang & Wu 2001) has extended the applicability of ant systems further into continuous search space.

1.3 Thesis Structure

There are seven main chapters in this thesis. Chapter 2 reviews the history of k-medoids algorithms and introduces one of the typical partitioning algorithms – k-medoids-based algorithm for the reader. It includes the complexity of three well-known algorithms–PAM, $CLARA$ and $CLARANS$. The concept of Fuzzy k-means clustering algorithm and genetic k-medoids algorithm are also addressed in this chapter. An improved k-medoids algorithm using simulated annealing termed $CLASA$ which has been developed and compared with other k-medoids algorithms. In Chapter 2, a novel and efficient approach is proposed to reduce the computational complexity of such k-medoids-based algorithms by using previous medoid index, triangular inequality elimination criteria, and partial distance search. In addition, the concept of the utilization of memory for efficient medoid search is presented. The proposed hybrid search approach which combines the previous medoid index, the utilization of memory, the criterion of triangular inequality elimination and the partial distance search for the problem of nearest neighbor search will be described clearly in this chapter.

Chapter 3 details a novel sampling scheme using multi-centroids with multi-

runs ($MCMRS$). This sampling scheme can be applied to PAM, $CLARA$, $CLARANS$ and $CLASA$ algorithms. A $MCMRS$-$CLASA$ algorithm that combines the benefits of $MCMRS$ with the $CLASA$ algorithm is also proposed and evaluated. The computation load in $MCMRS$ and $MCMRS$-$CLASA$ can be further released by applying a more efficient centroid-based clustering method. A more advanced sampling scheme termed the *Incremental Multi-Centroid, Multi-Run Sampling Scheme* ($IMCMRS$) is also presented in this chapter.

In contrast to Chapter 2, Chapter 4 explores in depth work done in the centroid-based clustering algorithms. A cluster generation algorithm for vector quantization using tabu search approach with simulated annealing is proposed. We also introduce the concept of *VQ*, genetic algorithms, tabu search and simulated annealing and describe our hybrid algorithm applying to clustering. The main idea of this algorithm is to use the tabu search approach to generate non-local moves for the clusters and apply the simulated annealing technique to select suitable current best solution to improve the speed of the cluster generation and reduce the mean squared error of the clustering. A novel watermarking algorithm based on labeled bisecting k-means clustering technique is presented in Chapter 4. The embedding process is performed by assigning the input vector to the cluster whose label is equal to the watermark bit. The related details will be described in Chapter 4. An efficient nearest neighbour codeword search algorithm based on *Hadamard* transform for applying to *VQ* is also introduced in this chapter. Four efficient elimination criteria are derived from two important inequalities based on three characteristic values in the *Hadamard* transform domain. Experimental results demonstrate the proposed algorithm is much more efficient than most existing nearest neighbour codeword search algorithms in the case of high dimension.

The swarm intelligence algorithms will be introduced in Chapters 5 and 6. Firstly, the particle swarm optimization is presented in Chapter 5. A parallelised version of the particle swarm optimization scheme is performed. Three

communication strategies for $PPSO$ are discussed, which can be used according to the strength of the correlation of parameters. In Chapter 6, two well known swarm intelligence algorithms–ant systems (AS) and ant colony systems (ACS) are introduced. We also proposed the Parallel Ant Colony Systems ($PACS$) and Parallel Ant Systems (PAS) to improve the performance for the problem of AS and ACS. Rather a parallel formulation is developed which gives not only reduces the computation time but also obtains a better solution. Pioneer works on traveling salesman problem using Parallel Ant Colony Systems and Parallel Ant Systems are explored in this chapter.

The final chapter concludes the thesis with future work and a summary of contributions.

Chapter 2

Improved K-medoids Algorithms

Various clustering algorithms have been designed to fit various requirements and constraints of application. In this chapter a number of improvements are suggested that can be applied to most k-medoids-based algorithms. These can be divided into two categories – conceptual / algorithmic improvements, and implementational improvements. These include the revisiting of the accepted cases for swap comparison and the application of *partial distance searching* and *previous medoid indexing* to clustering. We propose extensions to the problem of nearest neighbor search, by combining the *previous medoid index* with *triangular inequality elimination* and *partial distance searching*. An improved k-medoids algorithm using simulated annealing, $CLASA$, is also discussed, as is a novel mechanism for managing memory usage.

2.1 Related existing K-Medoids Algorithms

Significantly, no single algorithm is suitable for all types of object, nor are all algorithms appropriate for all problems, however, k-medoids algorithms have been shown to be robust to outliers and are not generally influenced by the order of presentation of the objects. Moreover, k-medoids algorithms are invariant to

translation and the orthogonal transformation of objects (Kaufman & Rousseeuw 1990). Both k-means (which adopts as the representative point the weighted mean of the cluster) and k-medoids (which adopts as the representative point the most central object in the cluster) algorithm are partitioning methods.

If we assume T objects $x_1, x_2, \ldots x_T$ and k objects chosen from these T objects as the representative objects (medoids) $o_1, o_2, \ldots o_k$ then the total distance for partitioning these $(T-k)$ objects based on the k representative objects is

$$D_t = \sum_{x_m \in S_p} d(x_m, o_p) \tag{2.1}$$

where S_p is the pth partitioned set (or cluster) such that $d(x_m, o_p) \leq d(x_m, o_n)$, $n = 1 \ldots k$, and $m = 1 \ldots T$. Clearly, this is a combinatorial optimization problem of choosing k medoids from T objects, and as such there are $\frac{T!}{k!(T-k)!}$ combinations. This is an NP-hard problem and complexity can be high for even for a moderate number of objects.

There are a number of existing k-medoids-based algorithms including Partitioning Around Medoids (PAM) (Kaufman & Rousseeuw 1990), Clustering LARge Applications ($CLARA$) (Kaufman & Rousseeuw 1990), Clustering Large Applications based on RANdomized Search ($CLARANS$) (Ng & Han 2002), Fuzzy c-medoids algorithm (Krishnapuram et al. 1999), Fuzzy c-trimmed medoids algorithm (Krishnapuram et al. 1999) and a genetic k-medoids algorithm (Lucasius, Dane & Kateman 1993) while the Clustering Large Applications based on Simulated Annealing ($CLASA$) algorithm applies simulated annealing (Kirkpatrick et al. 1983, Huang, Pan, Lu, Sun & Hang 2001) to select better medoids (Chu, Roddick & Pan 2001). The major computational drawback of k-medoids algorithms is their time complexity in discovering the medoids. In this chapter, the $CLASA$, the three efficient versions of $CLASA$ and three extended versions of $CLARANS$ (as an exemplar k-medoid-based algorithm) are presented based on the proposed efficiencies for search algorithms.

One of the goals in terms of improving clustering efficiency is to reduce the

number of distance calculations required to determine medoids. In this respect, the complexity of various routines can be compared by evaluating the number of distances needing to be calculated. This approach eliminates implementation and dataset considerations. In this section we discuss the complexity of three well-known algorithms, PAM, $CLARA$ and $CLARANS$, as examples of the wider class of algorithms that can benefit from our proposals.

2.1.1 PAM - Partitioning Around Medoids

The PAM k-medoids clustering algorithm, for example, evaluates a set of k objects considered to be representative objects (medoids) of k clusters within T objects such that the non-selected objects are clustered with the medoid to which it is the most similar (i.e. closest in terms of the provided distance metric). The process operates by swapping one of the medoids with one of the objects iteratively such that the total distance between non-selected objects and their medoid is reduced. The algorithm can be depicted as follows:

Step 1: Initialization - choose k medoids from T objects randomly.

Step 2: Evaluation - calculate the cost $D'_t - D_t$ for each swap of one medoid with one object, where D_t is the total distance before the swap and D'_t is the total distance after the swap.

Step 3: Selection - accept the swap with the best cost and if the cost is negative, go to step 2; otherwise record the medoids and terminate the program.

The computational complexity of the PAM algorithm is $O((1+\beta)k(T-k)^2)$ which is based on the number of partitions per object, where β is the number of successful swaps. It can also be expressed as $O'((1+\beta)k^2(T-k)^2)$ based on the number of distance calculations, i.e., one partition per object is equivalent to

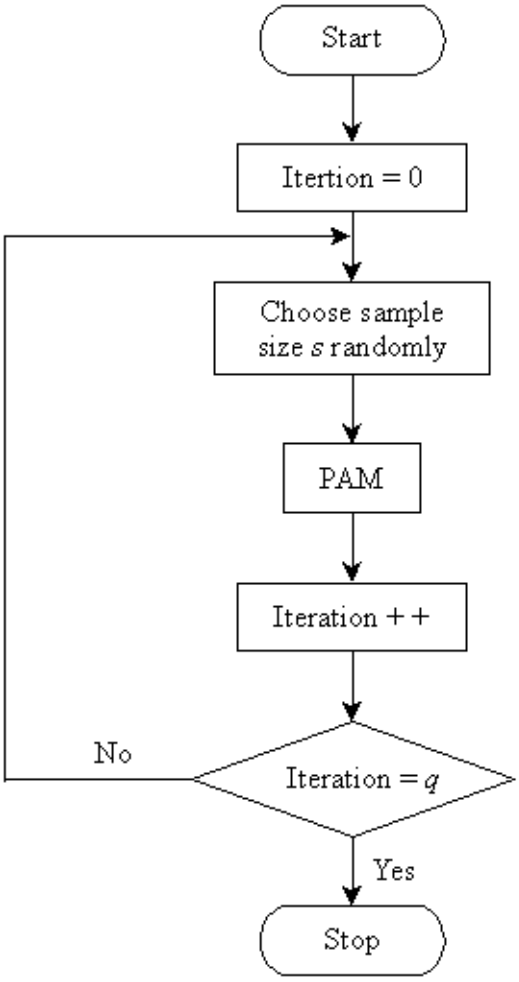


Figure 2.1. Clustering LARge Applications (*CLARA*)

k distances calculations. Clearly, this is time consuming even for the moderate number of objects and a small number of medoids.

2.1.2 *CLARA* – Clustering LARge Applications

CLARA (Clustering LARge Applications) (Kaufman & Rousseeuw 1990) shown in Figure 2.1 reduces the computational complexity by drawing multiple samples of the objects and applying the *PAM* algorithm on each sample. The final medoids are obtained from the best result of these multiple passes as below:

Repeat the following steps q times

Step 1: Call the PAM algorithm with a random sample, s objects from the original set of T objects.

Step 2: Partition the T objects based on the k medoids obtained from previous step. Update the better medoids based on the average distance of the partition.

The computational complexity of the $CLARA$ algorithm is $O(q(ks^2 + (T - k)) + \beta ks^2)$ based on the number of partitions per object or $O'(q(k^2s^2 + k(T - k)) + \beta k^2 s^2)$ based on the number of distance calculations, where q, s, k, β and T are the number of samples, object size per sample, number of medoids, the number of successful swaps for all samples tested and the total number of objects, respectively. Clearly, the $CLARA$ algorithm can deal with a larger number objects than can PAM algorithm if $s \ll T$.

If the sample size s is not large enough, the effectiveness (ie. the average distance) of the $CLARA$ algorithm is reduced. However, the efficiency (in terms of computation time) is impaired if the sample size is too large. There is tradeoff between the effectiveness and efficiency in $CLARA$ algorithm. The best clustering cannot be obtained in $CLARA$ if one of the best medoids is not included in the sample objects. In order to achieve both efficiency and acceptable performance (in terms of average distance per object) the $CLARANS$ (Clustering Large Applications based on RANdomized Search) algorithm (Ng & Han 2002) was proposed.

2.1.3 *CLARANS* – Clustering Large Applications Based on Randomized Search

The clustering process in *CLARANS* (Ng & Han 2002) is formalized as searching through a certain graph where each node is represented by a set of k medoids in which two nodes are neighbors if they differ by one medoid. Each node has $k(T-k)$ neighbors, where T is the total number of objects. *CLARANS* starts with a randomly selected node. It moves to a neighbour node if a test for the *maxneighbour* number of neighbours is successful; otherwise it records the current node as a local minimum. If the node is found to be a local minimum, it restarts with a new randomly selected node and repeats the search for a new local minimum. The procedure continues until some threshold *numlocal* of local minima have been found, and returns the best node. As shown in Figure 2.2, the *CLARANS* algorithm can be summarised as below:

Repeat the following steps *numlocal* times.

Step 1: Select a current node randomly and calculate the average distance of this current code, where node is the collection of k medoids.

Step 2: Repeat the following *maxneighbour* times.

- Select a neighbour node randomly and calculate the average distance for this node. If the average distance is lower, set current node to be the neighbour node.

The computational complexity is $O((\beta + numlocal)(T-k))$ based on the number of partitions per object or $O'((\beta + numlocal)k(T-k))$ based on the number of distance calculations, where β is the number of test moves between nodes.

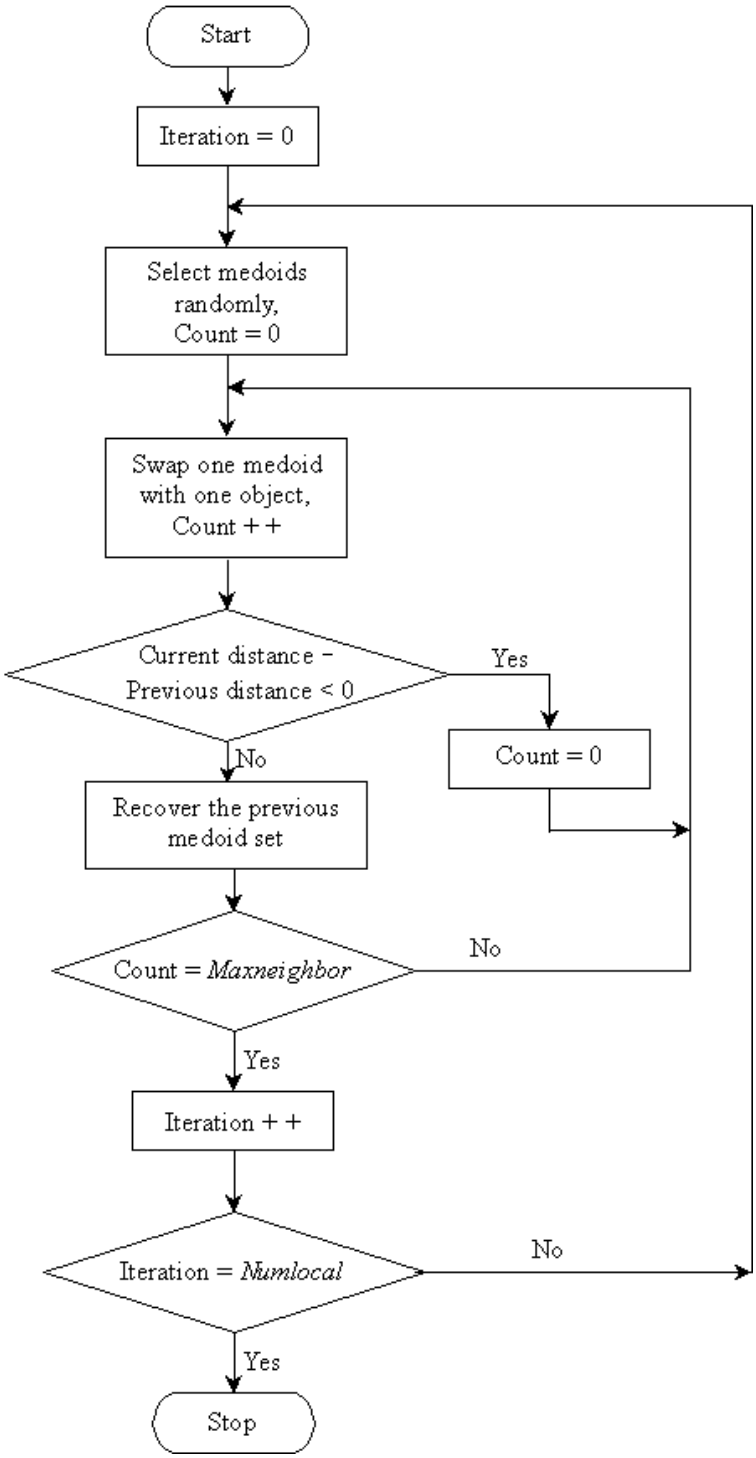


Figure 2.2: Clustering Large Applications Based on RANdomized Search Algorithm ($CLARANS$)

2.1.4 Fuzzy K-Medoids Algorithms

The concept of Fuzzy k-means clustering algorithm has been applied to generate the medoids. Two methods, the Fuzzy k-medoids algorithm and Fuzzy k Trimmed medoids algorithm were proposed in (Krishnapuram et al. 1999). The evaluation function of the Fuzzy k-medoids algorithm is to minimize

$$J_m(O, X) = \sum_{j=1}^{T} \sum_{i=1}^{k} u_{ij}^m r(x_j, o_i) \tag{2.2}$$

where $r(x_j, o_i)$ is the dissimilarity between object x_j and medoid o_i.

The Fuzzy k-medoids algorithm can be described as follows:

Step 1: Select initial set of medoids

$O = \{o_1, o_2, \ldots, o_k\}$ from the set of objects

$X = \{x_1, x_2, \ldots, x_T\}$ randomly, where k is the number of clusters and T is the number of objects.

Step 2: Calculate the membership using

$u_{ij} = \frac{\left(\frac{1}{r(x_j, o_i)}\right)^{\frac{1}{m-1}}}{\sum_{q=1}^{k} \left(\frac{1}{r(x_j, o_q)}\right)^{\frac{1}{m-1}}}$, $i = 1, \ldots, k$, $j = 1, \ldots, T$. where $r(x_j, o_i)$ denotes the dissimilarity between object x_j and medoid o_i , $m \in [1, \infty]$ is the fuzzifier.

Step 3: Update the new medoids using

$p = argmin_{1 \le v \le T} \sum_{j=1}^{T} u_{vj}^m r(x_v, x_j)$,

$o_i = x_p$, $i = 1, \ldots, k$.

Step 4: The program is terminated if the medoids are unchanged or the maximum number of iterations is reached.

A more robust fuzzy k-medoid algorithm can be obtained by using the idea of Least Trimmed Squares (Kim, Krishnapuram & Dave 1996). Substituting the

expression of membership u_{ij} into $Eq.$ 2.2, the evaluation function

$$J_m(O,X) = \sum_{j=1}^{T} \Big(\sum_{i=1}^{k} (r(x_j, o_i))^{\frac{1}{1-m}} \Big)^{1-m} = \sum_{j=1}^{T} h_j \tag{2.3}$$

where $h_j = \big(\sum_{i=1}^{k} (r(x_j, o_i))^{\frac{1}{1-m}} \big)^{1-m}$.

The evaluation criterion for the Fuzzy k Trimmed medoids algorithm is as follows:

$$J_m^T(O,X) = \sum_{q=1}^{s} h_q \tag{2.4}$$

The Fuzzy k-trimmed medoids algorithm can be described as follows:

Step 1: Select initial set of medoids

$O = \{o_1, o_2, \ldots, o_k\}$ from the set of objects

$X = \{x_1, x_2, \ldots, x_T\}$ randomly, where k is the number of clusters and T is the number of objects.

Step 2: Calculate the harmonic dissimilarities

$h_j = \big(\sum_{i=1}^{k} (r(x_j, o_i)) \frac{1}{1-m} \big)^{1-m}$, $j = 1, \ldots, T$

sort h_j in the ascending order such that

$h_1 \leq h_2 \leq \ldots \leq h_T$.

Step3: Compute the memberships for s objects, $u_{ij} = \dfrac{\Big(\frac{1}{r(x_j, o_i)}\Big)^{\frac{1}{m-1}}}{\sum_{q=1}^{k} \Big(\frac{1}{r(x_j, o_q)}\Big)^{\frac{1}{m-1}}}$,

$i = 1, \ldots, k, j = 1, \ldots, s$.

Step 4: Update the new medoids using

$p = argmin_{1 \leq v \leq s} \sum_{j=1}^{s} u_{vj}^m r(x_v, x_j)$,

$o_i = x_p$, $i = 1, \ldots k$.

Step 5: The program is terminated if the medoids are unchanged or the maximum number of iterations is reached.

2.1.5 Genetic K-Medoids Algorithm

Genetic algorithm (Goldberg 1989) has been applied to k-medoids algorithm (Lucasius et al. 1993). In (Lucasius et al. 1993), the genetic k-medoids algorithm is called GCA (*genetic clustering algorithm*). The basic idea of GCA is to generate P individuals initially and each individual consists of k different parameters selected from T objects randomly. The fitness function is the inverse of the total distance. The fitness is also modified using a linear scaling technique. The modified fitness of each individual is evaluated and pair of individuals is selected based on the roulette selection. These two selected individuals are used for crossover operation to generate temporary individual with half the parameters from each selected individual. If the temporary individual is a wrong one, i.e., two parameters are the same, then simply replace by one of the selected individual. The mutation technique is applied to this temporary individual. After getting the same population size, the evaluation, selection, crossover and mutation are applied again until the maximum number of generations is reached or the satisfied fitness is obtained. The experiments are carried out to test the performance of GCA and $CLARA$ algorithms. Experimental results demonstrate the GCA is superior to $CLARA$ for a large number of medoids. For small number of medoids, both $CLARA$ and GCA find acceptable solutions. The computational complexity of GCA based on the number of distance calculation is $O'(PGk(T-k))$, where G is the number of generations.

2.2 $CLASA$ - Clustering Large Applications Based on Simulated Annealing

Simulated annealing (Kirkpatrick et al. 1983) is a random search method that has been presented for NP-hard problems. Vecchi and Kirkpatrick (Vecchi & Kirkpatrick 1983) employed simulated annealing to optimize the wiring problem

and Gamal *et al.* used simulated annealing to construct good source codes, error-correcting codes and spherical codes (Gamal et al. 1987). Simulated annealing was also applied to codebook design for vector quantization (Huang, Pan, Lu, Sun & Hang 2001, Cetin & Weerackody 1988).

2.2.1 What's $CLASA$

A new approach using simulated annealing for selecting the medoids is presented in this section. In the $CLASA$ algorithm (Chu et al. 2001), we term the different collections of k-medoids the *state* – there are $\frac{T!}{k!(T-k)!}$ states. It is possible to move from current state to any other states depending on the moving strategy. For the preliminary experiments, we may consider only move the current state to the next state by only changing one medoid. As shown in Figure 2.3, the $CLASA$ algorithm can be illustrated as follows:

Step 1: Choose an initial state s of the medoids at random and set the initial temperature $Temp = T_0$.

Step 2: Randomly choose another state s' (perturbation of state s) by swapping the medoids with the objects. Calculate the difference of total distance $\Delta D = D_t(s') - D_t(s)$. If $\Delta D < 0$, replace the state s by s'; otherwise replace s by s' with probability $e^{\frac{-\Delta D}{Temp}}$.

Step 3: If the times of total distance drops $Threshold - drop$ exceeds a prescribed number or the fixed times of perturbations $Threshold - per$ is reached, go to step 4; otherwise go to step 2.

Step 4: Terminate the program and return the selected medoids if the temperature $Temp$ is below some prescribed freezing temperature T_f or the total times of perturbation $TotalPer$ is reached; otherwise lower the temperature $Temp$ and go to step 2.

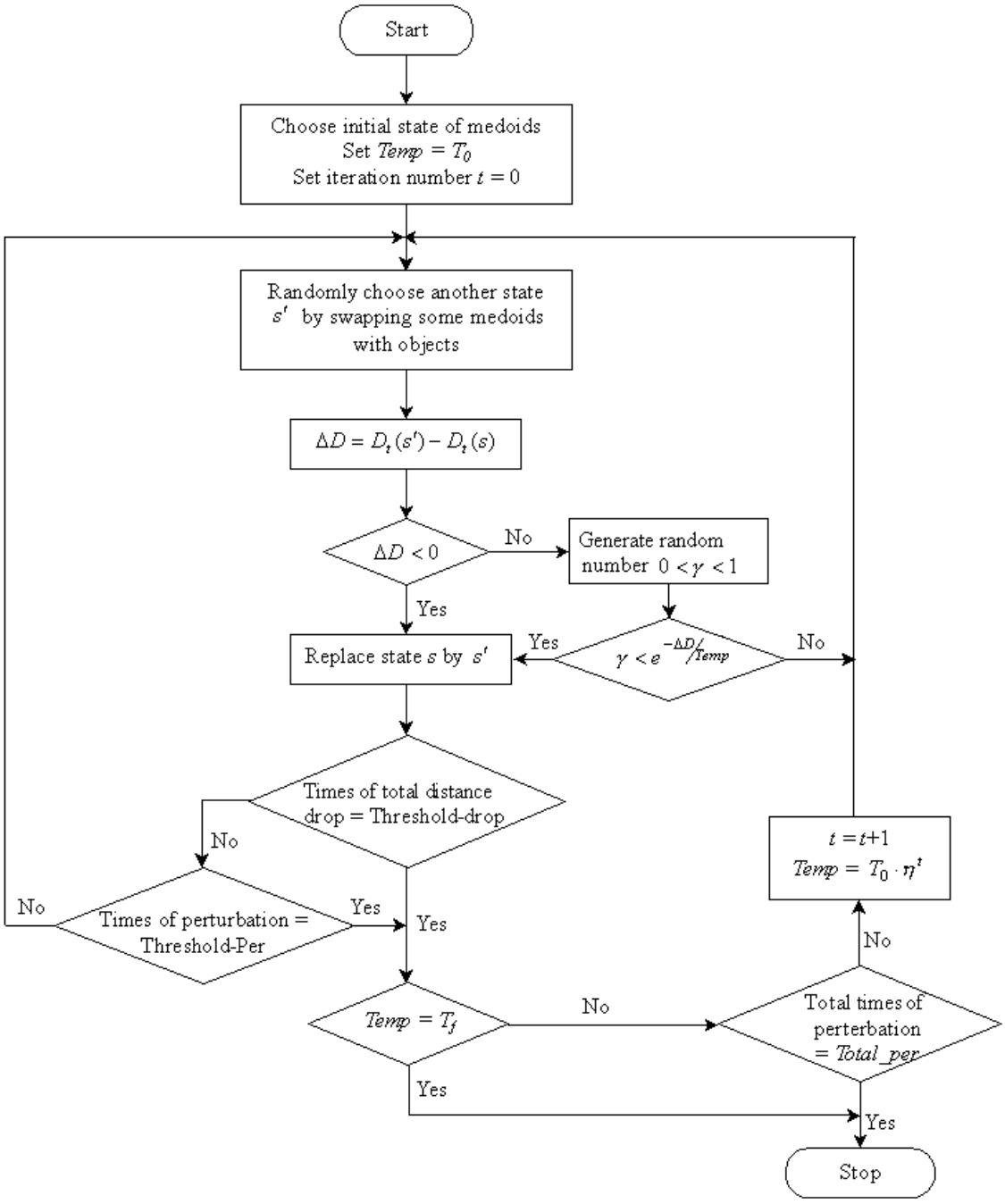


Figure 2.3: Clustering Large Applications Based on Simulated Annealing Algorithm ($CLASA$)

There are several possible methods for the annealing schedule, it is convenient to set $Temp = T_0\eta^t$, where t is the number of iterations, η is a constant coefficient, $0 < \eta < 1$.

2.2.2 Experiments

Four artificial databases are used for the experiments as follows:

1. 400 objects with 8 dimensions are generated from the Gaussian source with zero mean and unit variance.

2. 1000 objects with 8 dimensions are generated from the Gauss-Markov source which is of the form $y_n = \alpha y_{n-1} + w_n$ where w_n is a zero-mean, unit variance, Gaussian white noise process, with $\alpha = 0.5$.

3. 3000 objects collected from 20 rectangular clusters and 150 objects with two dimensions belonged to each cluster. The object generation program is in Figure 2.4.

4. 1500 objects collected from elliptic clusters as shown in Figure 2.5.

Experiments were carried out to test the number of distances calculation and the average distance per object for PAM, $CLARA$, $CLARANS$ and $CLASA$ algorithms. Squared Euclidean distance measure is used. For the first three experiments in the $CLARA$ algorithm, the parameter q was set to 5 and s was set to $40 + 2 * k$ for the sample size, where k is the number of medoids. The first experiment was to use Gaussian source with zero mean and unit variance to do the experiment. The total number of objects is 400 and the number of dimensions is 8 for each object. 20 medoids are selected from these 400 objects. For $CLARANS$ algorithm, the parameters *numlocal* and *maxneighbor* were set to 30 and 50, respectively. For the $CLASA$ algorithm, the parameters for the final temperature T_f, the initial temperature T_0, η, the number of total distance

```
void object_generation( ) {
#define num_point_cluster 150
#define division_x 5
#define division_y 4
int j,k,i,num; float object[division_x*division_y*num_point_cluster][2];
for (j=0;j<division_x;j++){
   for (k=0;k<division_y;k++){
      for (i=0;i<num_point_cluster;i++){
         num = i+(j*division_y+k)*num_point_cluster;
         object[num][0]=RandVal((float)j,(float)j+1.0);
         object[num][1]=RandVal((float)k,(float)k+1.0); }}}
 float RandVal (float low, float high)
{float val;
   val=((float) (rand() % 1000) / 1000.0) * (high - low) + low;
   return (val);}
```

Figure 2.4. Data Generation Program

drop *disdrop*, *per* and the total number of perturbations *totalper* were set to 0.0005, 10, 0.95, 10, 80 and 250000 respectively. As shown in Table 2.1, with these settings $CLASA$ performed better than $CLARA$ both in the computation time and the average distance per object. For the similar number of distances calculation, $CLASA$ reduced the average distance for more than 4% compared with $CLARANS$. PAM is computationally expensive. For the similar average distance, $CLASA$ may reduce the computation time for more than 95%.

The rectangular clusters are used in the second experiment. 20 medoids are selected from 3000 objects. For $CLARANS$ algorithm, the parameters numlocal and maxneighbor are set to 50 and 200, respectively. For the $CLASA$ algorithm, the parameters for the final temperature T_f, the initial temperature T_0, η, the number of total distance drop *disdrop*, *per* and the total number of perturbations *totalper* were set to 0.000025, 10, 0.95, 10, 80, 500000, respectively. As shown in Table 2.2, $CLASA$ performs better than $CLARANS$ both in the computation time and the average distance. Since the performance of $CLARANS$ is much better than both PAM and $CLARA$ presented in (Ng & Han 1994) using the

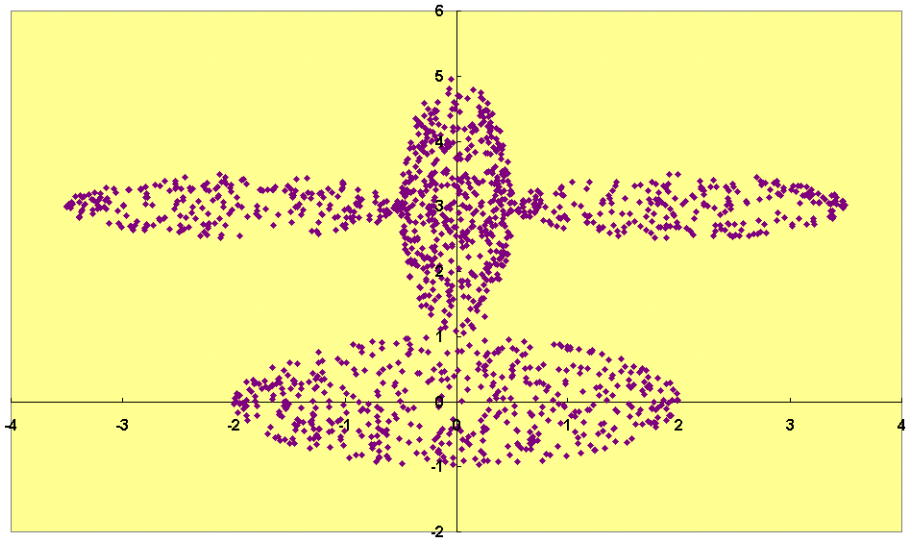


Figure 2.5. Elliptic Clusters

similar database, we did not tune the computation time and average distance for both PAM and $CLARA$ for the purpose of comparison. Actually, the performance of $CLASA$ is better than PAM and $CLARA$ for the same running time.

The Gauss-Markov source was used for the third experiment. 30 medoids are selected from 1000 objects. For $CLARANS$ algorithm, the parameters numlocal and maxneighbor are set to 30 and 50, respectively. For $CLASA$ algorithm, the parameters for the final temperature T_f, the initial temperature T_0, η, the number of total distance drop *disdrop*, *per* and the total number of perturbations *totalper* were set to 0.000025, 10, 0.95, 10, 80, 500000, respectively.

As shown in Table 2.3, $CLASA$ can reduce the computation time by 33% and decrease the average distance per object for 4.76% by comparing with $CLARANS$. $CLASA$ can also reduce the computation time by 34% and decrease the average distance per object for 13% in comparison with $CLARA$. Experimental results confirm the usefulness of $CLASA$.

The Gauss-Markov source was also used for the fourth experiment and 20 medoids were selected from 1000 objects. For $CLARA$ algorithm, the parameter

Table 2.1. Results of Experiment for Gaussian Source

seed	*PAM*		*CLARA*		*CLARANS*		*CLASA*	
	Average dist.	Count of dist.(10^5)	Average dist.	Count of dist.(10^5)	Average dist.	Count of dist.(10^5)	Average dist.	Count of dist.(10^5)
1	4.32	15007	4.75	851	4.61	762	4.38	662
2	4.30	14433	4.72	779	4.64	673	4.32	664
3	4.31	16155	4.60	880	4.50	664	4.36	666
4	4.33	12138	4.70	807	4.59	648	4.35	659
5	4.34	12138	4.71	721	4.57	697	4.35	662
6	4.30	17877	4.75	836	4.60	662	4.41	662
7	4.30	14433	4.83	908	4.50	764	4.40	660
8	4.32	18451	4.78	908	4.59	659	4.34	661
9	4.37	10990	4.79	836	4.50	654	4.37	662
10	4.30	16729	4.73	880	4.49	660	4.37	656
Ave.	4.32	14835	4.74	841	4.56	684	4.37	661

q was set to 5 and s was set to $2k + 40$, $2k + 80$, $2k + 160$ and $2k + 320$. For the $CLARANS$ algorithm, the parameter *numlocal* was set to 50 and parameter *maxneighbor* was set to 50, 100, 200 and 300. For $CLASA$ algorithm, the parameters for the initial temperature T_0, η and the number of total distance drop *disdrop* were set to 10, 0.95, and 10, respectively. The final temperature T_f was set to 0.0025, 0.00025, 0.000025 and 0.0000025. The parameter *per* was set to 100, 200, 500 and 1000. The total number of perturbations *totalper* was set to 5000, 50000, 500000 and 50000000. Experimental results are shown in Figure 2.6. Clearly, the proposed $CLASA$ algorithm can not only reduce computation time, but also the average distance per object.

The elliptic clusters were used for the fifth experiment and 12 medoids are selected from 1500 objects. For $CLARA$ algorithm, the parameter q was set to 5 and s was set to $2k + 160$. For the $CLARANS$ algorithm, the parameter *numlocal* was set to 5 and parameter *maxneighbor* was set to 270 (i.e. 1.5% of $k * T$). For $CLASA$, the parameters for the final temperature T_f, the initial temperature T_0, η, the number of total distance drop *disdrop*, *per* and the total number of perturbations *totalper* were set to 0.000025, 0.0001, 0.95, 10, 200, 50000, respectively. Experimental results are shown in Table 2.4. To enable

Table 2.2. Results of Experiment for Rectangular Clusters

seed	*PAM*		*CLARA*		*CLARANS*		*CLASA*	
	Average dist.	Count of dist.(10^5)	Average dist.	Count of dist.(10^5)	Average dist.	Count of dist.(10^5)	Average dist.	Count of dist.(10^5)
1	0.163	1640305	0.194	954	0.181	27922	0.167	26676
2	0.164	1333649	0.206	983	0.181	25534	0.167	27139
3	0.162	2265256	0.196	940	0.179	30487	0.170	27183
4	0.167	1796543	0.200	1084	0.180	29161	0.170	27181
5	0.167	1679364	0.217	1084	0.180	29538	0.171	25976
6	0.167	1679364	0.205	1069	0.186	26499	0.171	26466
7	0.163	2109018	0.196	1069	0.186	30026	0.171	25281
8	0.163	1679364	0.219	1041	0.182	30561	0.173	25793
9	0.163	1913721	0.205	925	0.177	28981	0.172	26990
10	0.164	2148098	0.209	1055	0.182	31592	0.170	26290
Ave.	0.164	1824468	0.205	1020	0.181	29030	0.170	26498

comparison, the experimental results of $CLARA$ and $CLARANS$ algorithms are averaged for 10 seeds and the result of the first seed of $CLASA$ algorithm are drawn in Figure 2.7. In our tests, the proposed $CLASA$ algorithm outperforms the $CLARA$ and $CLARANS$ algorithms.

Preliminary experiments using four artificial databases demonstrate that this proposed $CLASA$ algorithm not only may reduce the average distance but also improve the speed the clustering process comparing with respect to the PAM, $CLARA$ and $CLARANS$ algorithms.

2.3 Efficient Search Based K-Medoids Algorithms

A distance measure $D(X, C_i)$ is a non-negative dissimilarity measure between object X and medoids C_i. This distance is used to measure how close the input object X is to these medoids C_i. The nearest medoid is to be selected in order to encode the input object X. Therefore, encoding each input object requires N distance computations and $N - 1$ comparisons.

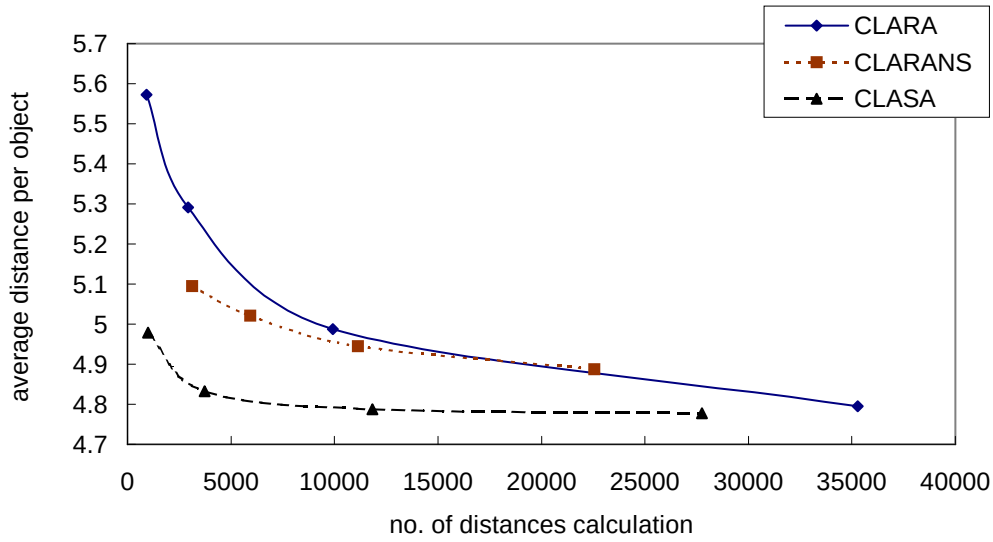


Figure 2.6: Performance Comparison of *CLARA*, *CLARANS* and *CLASA* for Gauss-Markov Sources

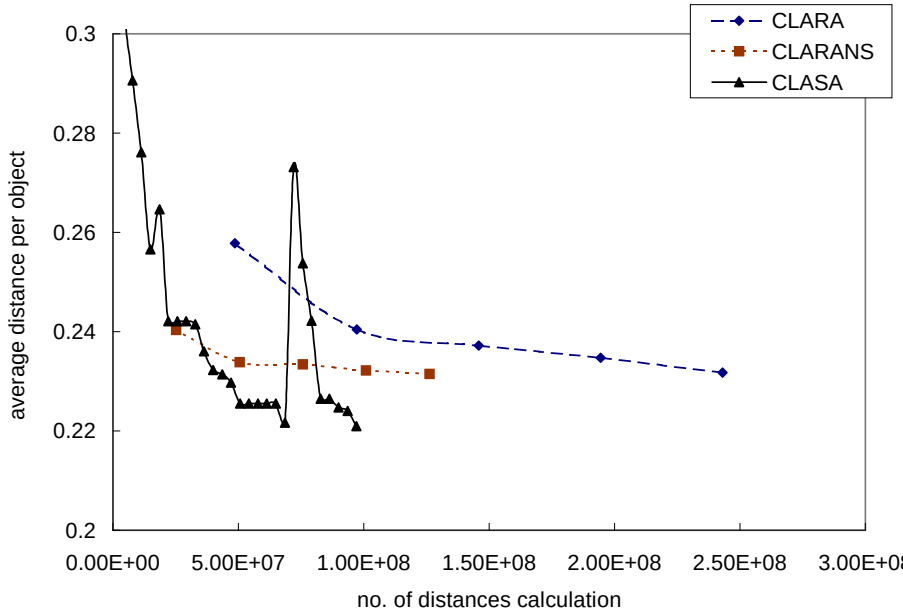


Figure 2.7: Performance Comparison of *CLARA*, *CLARANS* and *CLASA* for Elliptic Clusters

Table 2.3. Results of Experiment for Gauss-Markov Source

seed	*PAM*		*CLARA*		*CLARANS*		*CLASA*	
	Average dist.	Count of dist.(10^5)	Average dist.	Count of dist.(10^5)	Average dist.	Count of dist.(10^5)	Average dist.	Count of dist.(10^5)
1	4.061	293457	5.053	3355	4.618	3895	4.401	2404
2	4.053	326831	5.048	3399	4.552	3573	4.429	2439
3	4.066	335175	5.261	3532	4.613	3522	4.418	2376
4	4.038	318488	4.971	3266	4.673	3957	4.443	2404
5	4.045	343519	5.055	4153	4.733	3870	4.437	2434
6	4.027	326831	4.988	3710	4.606	3747	4.398	2376
7	4.036	293457	4.930	3576	4.621	3168	4.382	2385
8	4.050	385236	5.065	3665	4.610	3489	4.381	2416
9	4.037	368549	5.215	4020	4.634	3485	4.354	2461
10	4.042	376893	5.146	4065	4.597	3545	4.415	2441
Ave.	4.046	336844	5.073	3674	4.626	3625	4.406	2414

2.3.1 Revisiting Swap-Comparison of *PAM*

In general, in the *PAM* (Partitioning Around Medoids) algorithm (Kaufman & Rousseeuw 1990, Ng & Han 1994) four cases are examined to decide whether it is worth swapping any object with a medoid. We suggest that it is more reasonable and reliable to add a fifth case. Indeed, considering these five cases is equivalent to calculating the difference in total distance before and after swapping. Given an object o_{new} and a representative object o_{old}, the difference in total distance by swapping the o_{new} with o_{old} can be calculated according to the following cases:

First Case: In Case 1 (shown in Figure 2.8), the rectangle indicates that object x_j has been initially assigned to the cluster represented by medoid o_{old} while the oval indicates that the object x_j is *closer* (using whatever distance measure is appropriate) to the representative object o_m than to the new medoid o_{new}, where o_m is the second most similar representative object to x_j, i.e. $d(x_j, o_{new}) \geq d(x_j, o_m)$ and $d(x_j, o_{old}) \leq d(x_j, o_m) \leq d(x_j, o_n)$, $n = 1 \ldots k$ and $n \neq old, m$. Hence the contribution for object x_j to the cost of swapping the medoid $o_{old} \leftrightarrow o_{new}$ is $C_j = d(x_j, o_m) - d(x_j, o_{old})$.

Table 2.4. Results of Experiment for Elliptic Clusters

seed	*CLARA*		*CLARANS*		*CLASA*	
	Average dist.	Count of dist.(10^5)	Average dist.	Count of dist.(10^5)	Average dist.	Count of dist.(10^5)
1	0.228	2004	0.228	1609	0.221	971
2	0.221	2302	0.232	1067	0.224	991
3	0.236	2388	0.236	1351	0.235	965
4	0.230	2686	0.233	1154	0.217	976
5	0.227	2430	0.232	1572	0.222	993
6	0.237	2260	0.227	1223	0.224	947
7	0.232	2646	0.234	895	0.222	973
8	0.236	2516	0.231	1429	0.231	927
9	0.233	2345	0.231	1148	0.223	993
10	0.238	2728	0.231	1167	0.227	994
Ave.	0.232	2431	0.232	1140	0.225	973

Second Case: Assume o_{old} is the most similar representative object to object x_j, and o_m is the second most similar representative object to object x_j, i.e. $d(x_j, o_{old}) \leq d(x_j, o_m) \leq d(x_j, o_n)$, $n = 1 \ldots k$ and $n \neq old, m$. Then, if the object x_j is closer to object o_{new} than to the second most similar representative object o_m, i.e. $d(x_j, o_{new}) < d(x_j, o_m)$. Hence, the contribution for object x_j to the cost of the swap is $C_j = d(x_j, o_{new}) - d(x_j, o_{old})$.

Third Case: Assume the object x_j is more dissimilar from both the old representative object o_{old} and the new one o_{new} than from one of the other representative objects (such as medoid o_m), i.e. $d(x_j, o_m) < d(x_j, o_{new})$ and $d(x_j, o_m) \leq d(x_j, o_n)$, $m \neq old, n = 1 \ldots k$. Thus, the contribution for object x_j to the cost C_j of the swap is zero.

Fourth Case: For case. 4, the object x_j is closer to the cluster with the representative object o_m, but object x_j is more similar to object o_{new} than o_m, i.e. $d(x_j, o_{new}) < d(x_j, o_m)$ and $d(x_j, o_m) \leq d(x_j, o_n)$, $n = 1 \ldots k$ and $m \neq old$. Then, the contribution for object x_j to the cost of the swap is $C_j = d(x_j, o_{new}) - d(x_j, o_m)$.

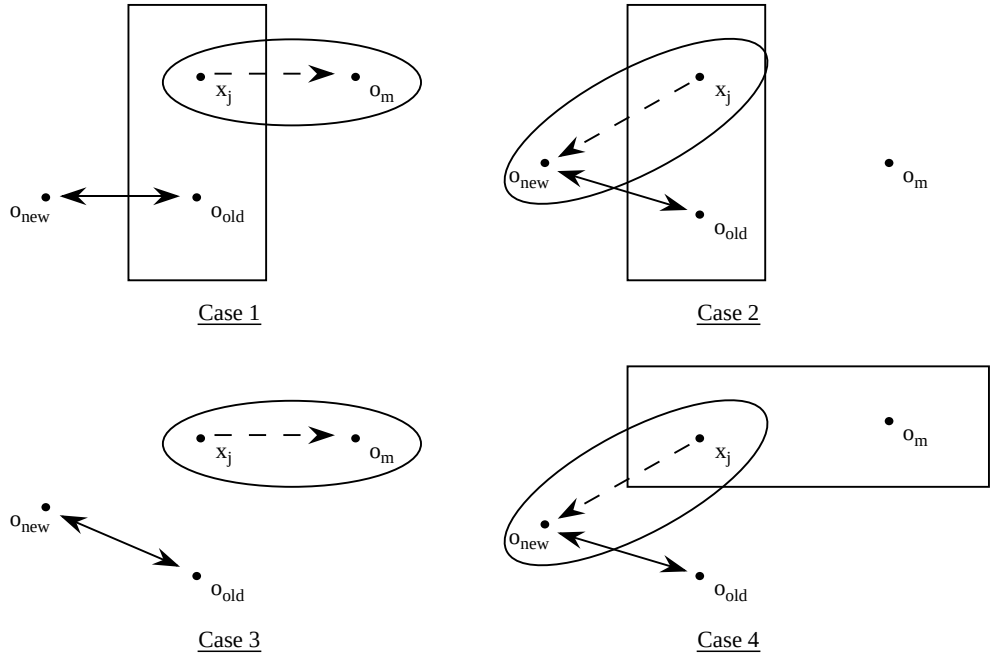


Figure 2.8: Four Cases When Medoid o_{old} Is Replaced as Representative Object by o_{new}

The above four cases (summarised as in Table 2.5) represent the effect of changes when medoids and non-medoid objects are swapped from the point of view of a non-medoid object. However, in the cases when the medoid object in question itself becomes the non-medoid object, (ie. in the cases above o_{old}) there is a fifth case to consider and which, if not considered, can result in algorithms that fail to terminate (Figure 2.9).

Table 2.5: Summarisation of Cases Regarding Cost of Replacing o_{old} with o_{new} as medoid

Order of Closeness to x_j	Result of Swap	Cost	Case
o_{old} - o_{new} - o_m	Cluster representation transfers from o_{old} to o_{new}	$d(x_j, o_{new}) - d(x_j, o_{old})$	2
o_{new} - o_{old} - o_m	Cluster representation transfers from o_{old} to o_{new}	$d(x_j, o_{new}) - d(x_j, o_{old})$	2
o_{old} - o_m - o_{new}	Cluster representation transfers from o_{old} to o_m	$d(x_j, o_m) - d(x_j, o_{old})$	1
o_{new} - o_m - o_{old}	Cluster representation transfers from o_m to o_{new}	$d(x_j, o_{new}) - d(x_j, o_m)$	4
o_m - $\left\{ \begin{matrix} o_{new} \\ o_{old} \end{matrix} \right\}$	Cluster representation remains with o_m	nil	3

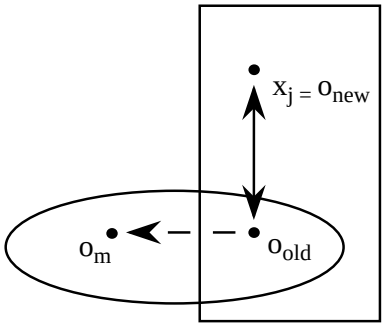


Figure 2.9: Fifth Case When Medoid o_{old} Becomes The Non-Medoid Object

Suppose the representative object o_m is the most similar medoid to o_{old} once o_{old} is swapped with x_j to become a new medoid o_{new}. As o_{old} is more similar to o_m than o_{new}, o_{old} will be assigned to the cluster represented by o_m, i.e. $d(o_{old}, o_{new}) > d(o_{old}, o_m)$ and $d(o_{old}, o_m) \leq d(o_{old}, o_n)$, $n = 1 \ldots k$ and $n \neq old, m$. Then, the contribution for o_{old} to the cost of the swap is

$$C_{old} = d(o_{old}, o_m) - d(o_{old}, o_{old}) = d(o_{old}, o_m) \tag{2.5}$$

Otherwise, if replacing o_{old} with o_{new} as a new representative object and o_{old} is less similar to o_m than o_{new}, o_{old} will belong to the cluster represented by o_{new}, i.e. $d(o_{new}, o_{old}) \leq d(o_{old}, o_m)$ and $d(o_{old}, o_m) \leq d(o_{old}, o_n)$, $n = 1 \ldots k$ and $n \neq old, m$. Hence, the contribution for o_{old} to the cost of the swap is

$$C_{old} = d(o_{old}, o_{new}) - d(o_{old}, o_{old}) = d(o_{new}, o_{old}) \tag{2.6}$$

The difference of total distance for the swap can thus be calculated by:

$$D'_t - D_t = C_{old} + \sum_j C_j \tag{2.7}$$

where D'_t is the total distance after the swap of the representative object o_{old} by object o_{new}, if the number of objects T is much larger than the number of representative objects k ($T \gg k$), then $D'_t - D_t \approx \sum_j C_j$. The cost of swapping

including the four cases is almost the same as to choose the difference of total distance as the cost. Since the representative object o_{old} also contributes the cost to the swap of the object o_{new} with o_{old}, it is more reasonable to consider the fifth case in the PAM algorithm. If $\sum_j C_j \geq 0$ for all pairs of medoids and objects, then the program is terminated. It is possible that the program never terminates (i.e. $\sum_j C_j$ is always negative), especially for small numbers of objects, due to C_{old} in the fifth case never being considered for the cost evaluation.

2.3.2 VQ-Based Techniques

Although k-medoids-based algorithms are designed for clustering large databases, all existing k-medoids-based algorithms can be time consuming when presented with significant volumes of data. The computational complexity of k-medoids-based algorithms can be improved by applying the concepts used in VQ-based codeword search (Bei & Gray 1985, Guan & Kamel 1992, Lee & Chen 1994, Baek, Jeon & Sung 1997, Pan, McInnes & Jack 1996*b*, Soleymani & Morgera 1987, Pan, Lu & Sun 2000, Vidal 1986, Chen & Pan 1989).

Vector Quantization (VQ) has been widely used for various real-time applications as encoding and recognition is an important factor in respect of the performance of these applications. Unfortunately, a full search is applied in VQ encoding and recognition and this is a time consuming process when the codebook size is large. A vector quantizer of rate r bits/sample and dimension k is a mapping from a k-dimensional vector space into some finite subset $C = \{C_j; j = 1, \ldots, N\}$, where $N = 2^{kr}$. The subset C is called a codebook and its elements C_j are called codewords, codevectors, reproducing vectors, prototypes or design samples. A distance measure $D(X, C_j)$ is a non-negative dissimilarity measure between vector X and codewords C_j. The distance is used to measure how close the input vector X is to these codewords C_j. The nearest codeword is selected in order to encode the input vector X. Therefore, encoding each input vector requires N

distance computations and $N-1$ comparisons.

The codeword search problem in VQ assigns a codeword to the test vector in which the distance between this codeword and the test vector is the smallest among all codewords. Given one codeword C_t and the test vector X in the k-dimensional space, the distance of the squared Euclidean metric can be expressed as follows:

$$D(X, C_t) = \sum_{i=1}^{k} (x^i - c_t^i)^2, \tag{2.8}$$

where $C_t = \{c_t^1, c_t^2, ..., c_t^k\}$ and $X = \{x^1, x^2, ..., x^k\}$.

Each distortion calculation requires k multiplications and $2k-1$ additions. Therefore, it is necessary to perform kN multiplications, $(2k-1)N$ additions, and $N-1$ comparisons to encode each input vector. The computational complexity depends on the codebook's size and dimensions. It needs a large codebook size and higher dimensionality for high performance in VQ encoding and recognition systems resulting in increased computation load during codeword search. In order to reduce the computational complexity of VQ encoding, a variety of techniques have been developed for codeword search. These algorithms can be grouped into three categories: spatial domain inequality based (Bei & Gray 1985, Vidal 1986, Chen & Pan 1989, Huang & Chen 1990, Huang, ChenBi, Stiles & Harris 1992, Guan & Kamel 1992, Ra & Kim 1993, Lee & Chen 1994, Baek et al. 1997, Pan & Huang 1998, Pan et al. 1996*c*, Pan et al. 1996*b*, Pan, Lu & Sun 2003, Wu & Lin 2000, Lu & Sun 2003), pyramid structure based (Lee & Chen 1995, Pan, Lu & Sun 2000, Song & Ra 2002*a*) and transform domain inequality based (Hwang, Jeng & Chen 1997, Lu, Pan & Sun 2000*a*, Jiang, Lu & Wang 2003).

The efficiencies of codeword search algorithms in VQ-based signal compression have never been applied to k-medoids-based algorithms. For example, the *equal-average nearest neighbor search* ($ENNS$) algorithm (Guan & Kamel 1992) uses the mean of an input vector to eliminate impossible codewords. This algorithm reduces a great deal of computation time in comparison with the conventional

full-search algorithm with only $O(k)$ additional memory, where k is the number of codewords (or the number of medoids for k-medoids-based algorithms). The improved algorithm (Lee & Chen 1994) uses the variance as well as the mean of an input vector. It can be referred to as the *equal-average equal-variance nearest neighbor search* ($EENNS$) algorithm. This algorithm reduces more computation time with double the additional memory of the $ENNS$ algorithm. The improved algorithm (Baek et al. 1997) presented by Baek *et. al.* using the mean and the variance of an input vector such as $EENNS$ to develop a new inequality between these features and the distance.

In (Pan et al. 1996*b*), the bound for Minkowski and quadratic metrics were derived and applied to codeword search. The partial distance search (PDS) (Bei & Gray 1985) and absolute error inequality criterion (AEI) (Soleymani & Morgera 1987) are all special cases in the bound for Minkowski metric. An inequality for fast codeword search based on the mean-variance pyramid was also derived (Pan, Lu & Sun 2000). We apply these concept to k-medoids-based algorithms.

2.3.3 Partial Distance Search

The partial distance search (PDS) algorithm (Bei & Gray 1985) is a simple and efficient codeword search algorithm which allows early termination of the distance calculation between a test vector and a codeword by introducing a premature exit condition in the search process. Given the squared Euclidean distance measure, one object $x = \{x_1, x_2, \ldots, x_d\}$ and two medoids (representative objects) $o_t = \{o_{t1}, o_{t2}, \ldots, o_{td}\}$ and $o_j = \{o_{j1}, o_{j2}, \ldots, o_{jd}\}$, if we assume the current minimum distance is

$$D(x, o_t) = \sum_{i=1}^{d} (x_i - o_{ti})^2 = D_{min} \tag{2.9}$$

$$if \qquad \sum_{i=1}^{h} (x_i - o_{ji})^2 \geq D_{min}, \tag{2.10}$$

$$then \qquad D(x, o_j) \geq D(x, o_t), \tag{2.11}$$

where $1 \leq h \leq d$. The efficiency of PDS is derived from the elimination of an unfinished distance computation if its partial accumulated distance is larger than the current minimum distance. This will reduce $(d-h)$ multiplications and $2(d-h)$ additions at the expense of h comparisons.

2.3.4 Triangular Inequality Elimination

Vidal proposed the approximating and elimination search algorithm (AESA) (Vidal 1986) whose computation time is approximately constant for a codeword search in a large codebook size. The high correlation characteristics between data vectors of adjacent speech frames and the triangular inequality elimination (TIE) criterion were utilized to VQ-based recognition of isolated words (Chen & Pan 1989). The triangular inequality elimination (TIE) criterion is an efficient method for applying to nearest neighbor search.

Let X be the set of data vectors and C be the set of codewords and x, y belong to the set X. Assume the distance measure existing for defining the mapping $d : X \times X \rightarrow R$, is used to fulfill the following metric properties:

$$d(x,y) \geq 0; d(x,y) = 0 iff x = y \tag{2.12}$$

$$d(x,y) = d(y,x) \tag{2.13}$$

$$d(x,y) + d(y,z) \geq d(x,z) \tag{2.14}$$

Let o_1 and o_2 be two different medoids and t be an object, then three TIE criteria can be obtained as following:

- Criterion 1 (shown in Figure 2.10):

  Given the triangular inequality

$$d(o_1,o_2) \leq d(t,o_2) + d(t,o_1) \tag{2.15}$$

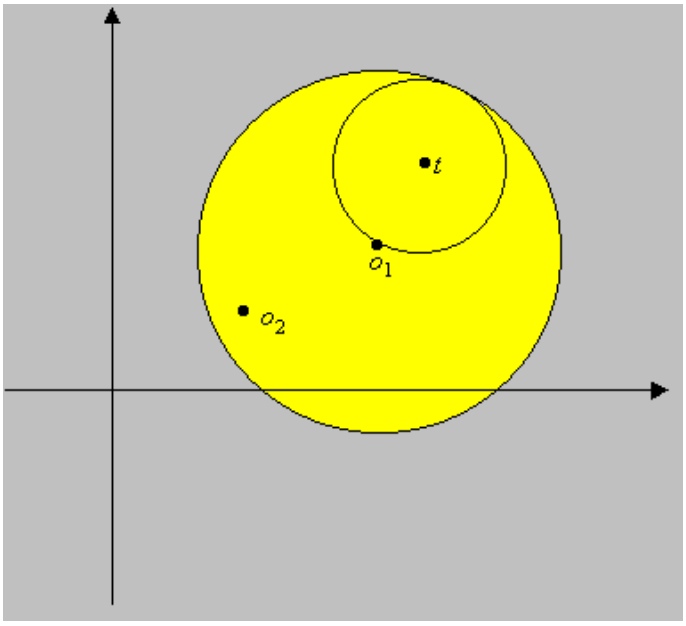


Figure 2.10. The First Criterion of TIE

if

$$d(o_1, o_2) \geq 2d(t, o_1) \tag{2.16}$$

then

$$d(t, o_2) \geq d(t, o_1) \tag{2.17}$$

Using this criterion, the distances between all pairs of medoids can be computed in advance. If Eq. 2.16 is satisfied, then we omit the computation of $d(t, o_2)$ if $d(t, o_1)$ has already been calculated.

In (Chu, Roddick & Pan 2002*b*), TIE criterion is modified for a squared error distance measure. Given the medoid size k, a table with memory size $k(k-1)/2$ is made to store one quarter of squared distance between medoids, ie. if

$$d^2(o_1, o_2)/4 \geq d^2(x, o_1) \tag{2.18}$$

then

$$d(x, o_2) \geq d(x, o_1) \tag{2.19}$$

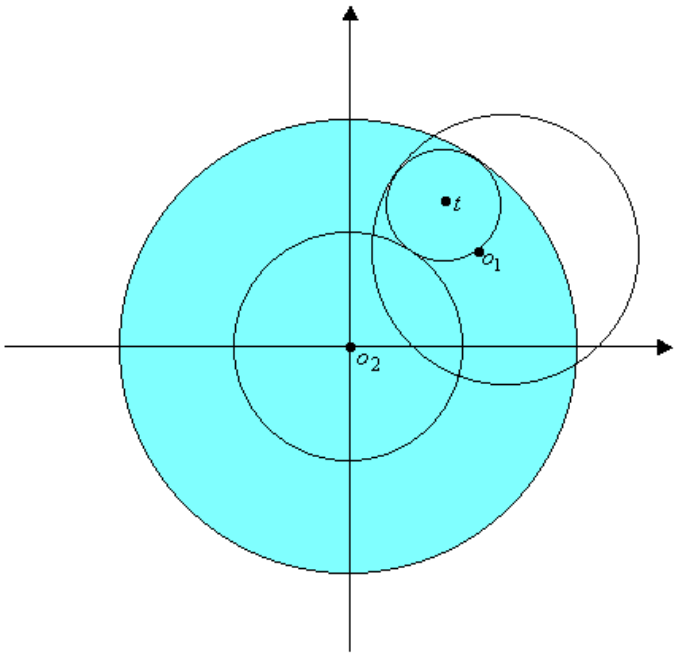


Figure 2.11. The Second Criterion of TIE

- Criterion 2 (shown in Figure 2.11):

  If the medoid $o_i, i \neq 1, 2$, does not locate centered on o_2 with radius $d(t, o_2)+ d(t, o_1)$, the computation of its distance to the test sample can be eliminated. Given the triangular inequality

  $$d(o_3, o_2) \leq d(t, o_2) + d(t, o_3); \tag{2.20}$$

  if

  $$d(o_3, o_2) \geq d(t, o_1) + d(t, o_2) \tag{2.21}$$

  then

  $$d(t, o_3) \geq d(t, o_1) \tag{2.22}$$

- Criterion 3 (shown in Figure 2.12):

  If the medoid $o_i, i \neq 1, 2$, is located centered on o_2 with radius $d(t, o_2) - d(t, o_1)$, the computation of its distance to the test sample can be omitted.

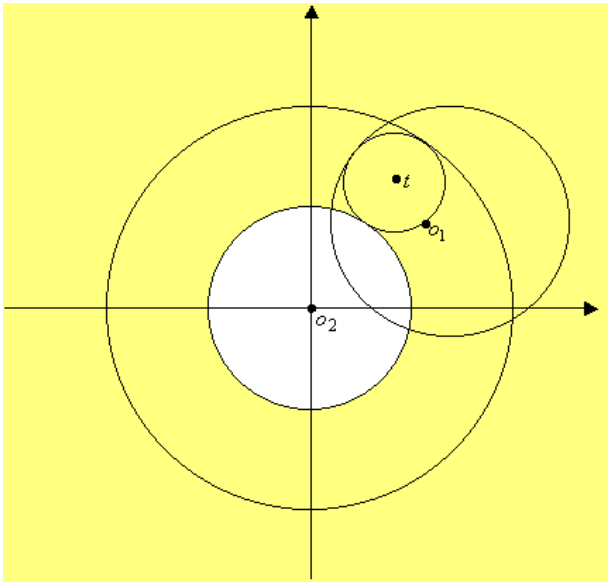


Figure 2.12. The Third Criterion of TIE

Assume $d(t, o_1) \leq d(t, o_2)$

Given

$$d(o_3, o_2) \geq d(t, o_2) - d(t, o_3); \tag{2.23}$$

if

$$d(o_3, o_2) \leq d(t, o_2) - d(t, o_1) \tag{2.24}$$

then

$$d(t, o_3) \geq d(t, o_1) \tag{2.25}$$

As shown in Figure 2.13, the physical meaning of combined Criterion 2 and 3 can be described as follows:

If the medoid $o_i, i \neq 1, 2$, does not locate between the two concentric circles centered on o_2 with radii $d(t, o_2) \pm d(t, o_1)$, the distance calculation to the test sample can be eliminated, i.e., if $d(o_1, o_2) > d(t, o_2) + d(t, o_1)$ or $d(o_1, o_2) < d(t, o_2) - d(t, o_1)$, then omit the distance computation of o_1 (Pan et al. 1996*c*). By merging criteria 2 and 3, we may get the following criterion, i.e.,

$$if \quad d(x, o_1) \leq |d(o_3, o_2) - d(x, o_2)| \tag{2.26}$$

$$then \quad d(x, o_3) \geq d(x, o_1) \tag{2.27}$$

Set

$$o_2 = \vec{0} \quad (zero\ vector).$$

Hence

$$if \quad d(x, o_1) \leq |d(o_3, \vec{o}) - d(x, \vec{0})|, \tag{2.28}$$

$$then \quad d(x, o_3) \geq d(x, o_1). \tag{2.29}$$

For Euclidean distance measure and given

$$d_{min} = d(x, o_1),$$

$$if \quad d_{min} \leq| \ |\sqrt{\sum_{i=1}^{d} o_{3i}^2} - \sqrt{\sum_{i=1}^{d} x_i^2} \ | \ |, \tag{2.30}$$

$$then \quad d(x, o_3) \geq d_{min}. \tag{2.31}$$

Since $\sqrt{\sum_{i=1}^{d} o_{3i}^2}$ can be calculated off line and $\sqrt{\sum_{i=1}^{d} x_i^2}$ is only computed once for the nearest neighbor search, the derived inequality ($Eq.2.28$–$Eq.2.31$) is very efficient for the problem of nearest neighbor search. Combine criterion 1, 2 and 3 shown in Figure 2.14, we can eliminate the most of distance computation if the medoid $o_i, i \neq 1, 2$ is located outside the gray area.

2.3.5 Previous Medoid Index

Most k-medoids-based algorithms check whether the designation of *medoid* needs to be transferred to another object. Since only one medoid is changed, most of the objects will continue to belong to the cluster represented by the same medoid. By using this property, we can calculate the distance between the object and its previous medoid index first. Since the probability is high that the object belongs to the same medoid index, the distance will tend to be small. If there is a small

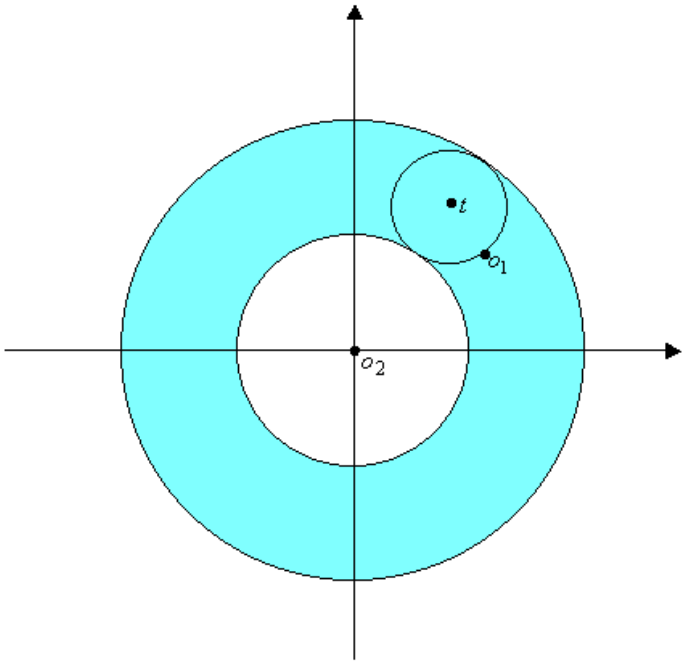


Figure 2.13. The Combination of Criterion 2 and 3 of TIE

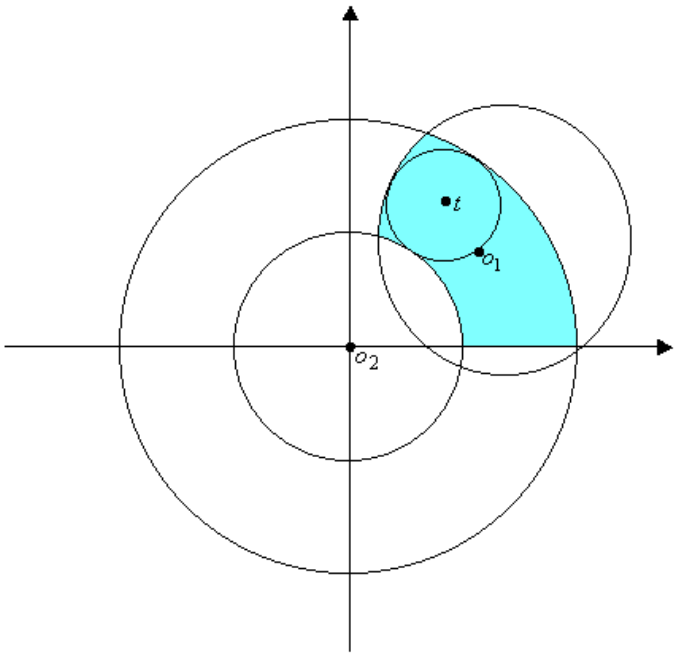


Figure 2.14. The Combination of Criterion 1, 2 and 3 of TIE

Before swap		After swap	
Object	Nearest Medoid	Object	Nearest Medoid
x_1	$NM(x_1)$	x_1	$NM(x_1)$ or o_{new}
x_2	$NM(x_2)$	x_2	$NM(x_2)$ or o_{new}
x_3	$NM(x_3)$	x_3	$NM(x_3)$ or o_{new}
$\vdots$	$\vdots$	$\vdots$	$\vdots$
x_h	$NM(x_h)$	o_{old}	o_1 or $o_2 \ldots o_k$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
x_{T-k}	$NM(x_{T-k})$	x_{T-k}	$NM(x_{T-k})$ or o_{new}

Figure 2.15. The Three Categories of Distance Calculation

distance between the object and one medoid, then it is easier to use the TIE and partial distance search criteria to reduce the distance computation.

2.4 Memory Utilization Based K-Medoids Algorithm

This section, discusses how the concepts outlined above can be efficiently accommodated into a k-medoids-based algorithm. Specifically, we present an approach to efficiently using the available memory and the modifications to $CLARANS$ used in the experiments (Chu, Roddick, Chen & Pan 2002).

Assume k medoids $o_j, j = 1, \ldots k$, are chosen from T objects $x_i, i = 1, \ldots T$, and the number of dimensions for each object or medoid is d. The size of the memory for all objects in the database is Td. If the distance table for each pair of objects $(x_i.x_j)$ is stored, then the memory requirement for the distance table is $\frac{T(T-1)}{2}$. If this memory is available, then all distance calculations need be performed just the once, whether for the PAM, $CLARA$ or $CLARANS$ algorithms. All these algorithms can thus be made more efficient and the computational complexity will be reduced.

Unfortunately, if the number of objects is large, memory is not always avail-

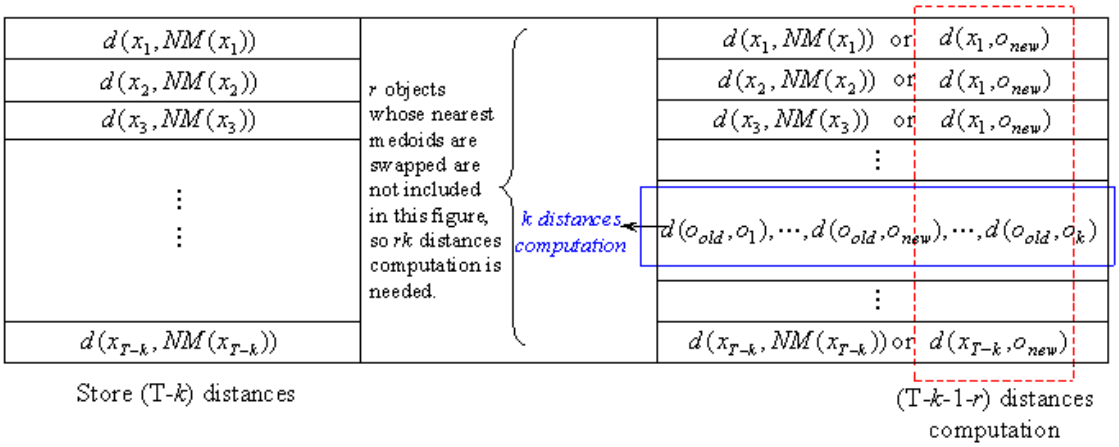


Figure 2.16. Usage of Memory for Distance Calculations

able. We thus propose a new approach which uses only $O(T-k)$ memory to store the distance, although it reduces from $O((T-k)k)$ to $O(T+rk)$ the number of distance computations required to test for the swap of two objects, where r is the number of objects whose nearest medoids are swapped. The probability of swapping the nearest medoid with any object is approximately $\frac{1}{k}$, so $r \approx \frac{T}{k}$ and thus the number of distance calculations can be reduced from $O(kT)$ to $O(T)$ if $T \gg k$. Assuming $NM(x_i)$ is the nearest medoid to the object x_i before the swap, the total distance before the swap of object o_{new} and medoid o_{old} can be expressed as

$$D_t = \sum_{i=1}^{T-k} d(x_i, NM(x_i)). \tag{2.32}$$

As is shown in Figure 2.15, the distance calculations for the swap of object o_{new} and medoid o_{old} can be separated into three categories. The utilisation of memory for the calculation of distance is shown in Figure 2.16.

First, for objects belonging to medoids not involved in the swap, the distance can be expressed as

$$D'_{t1} = \sum_{i=1}^{T-k} min(d(x_i, NM(x_i)), d(x_i, o_{new})). \tag{2.33}$$

Second, the distance for the medoid which is swapped to be an object is:

$$D'_{t2} = min(d(o_{old}, o_j)), j = 1, \ldots k. \tag{2.34}$$

Third, for those objects whose nearest medoids are swapped to be objects as follows:

$$D'_{t3} = \sum_{i=1}^{T-k} d(x_i, o_p), x_i \in S_p \tag{2.35}$$

where S_p is the p^{th} partitioned set where o_p is the representative medoid.

Hence the total distance after the swap of an object o_{new} and medoid o_{old} can be expressed as:

$$\begin{aligned} D'_t &= \sum_{i=1}^{T-k} \{min(d(x_i, NM(x_i)), d(x_i, o_{new})) \\ &+ d(x_i, o_p)\} \\ &+ min(d(o_{old}, o_j)), j = 1, \ldots k \end{aligned} \tag{2.36}$$

If the distances from all points to their medoids are stored (i.e. $d(x_i, NM(x_i))$, then only $(T - k - r - 1)$ distance calculations for $d(x_i, o_{new})$, k calculations for $d(o_{old}, o_j)$, and rk calculations for $d(x_i, o_p)$ are required. Since the memory size $(T - k)$ is generally reasonable for the clustering of objects with memory size Td, this is a useful approach. Note that this approach can be applied fairly widely including to clustering algorithms such as PAM, $CLARA$, $CLARANS$.

2.5 Experimental Results

In order to test the utility of the different approaches to the problem of medoid search, various combinations of the search elements were combined. Where appropriate, these new search approaches are applied to the $CLARANS$ and the proposed $CLASA$ algorithm and compared to $CLARA$ and $CLARANS$.

- Modified $CLARANS$ and $CLASA$ algorithms incorporating previous medoid index, the criterion of TIE and PDS. These algorithms are referred to as $CLARANS$-ITP and $CLASA-ITP$.

- $CLARANS$ and $CLASA$ with the proposed utilization of memory are referred to as $CLARANS$-M and $CLASA-M$.

- The application of the previous medoid index, the proposed utilization of memory, the criterion of TIE and partial distance search algorithm to $CLARANS$ and $CLASA$ are referred to as $CLARANS$-$MITP$ and $CLASA-MITP$.

Experiments were carried out to test the number of distance calculations, the running time and the average distance per object for $CLARA$, $CLARANS$, and the three extended version of $CLARANS$ and $CLASA$ above. Since computation time depends not only on the clustering algorithm but also on the choice of target system, it is better to choose a measurable system-independent criterion so that results are comparable – for clustering algorithms the number of distance calculations if often chosen. However, to provide some indication of real-world performance we also give our empirical run-time results which were run on an 850MHz Intel Pentium III. The squared Euclidean distance measure is used for the experiments. It can be noted that the number of distance calculations was approximately proportional to the computation time in all cases. Seven sets of experiments were conducted on different datasets as discussed below.

2.5.1 Gaussian Source

A dataset of 400 objects with 8 dimensions are generated from the Gaussian source with zero mean and unit variance to do the experiment. 20 medoids are selected from these 400 objects. For $CLARA$ algorithm, the parameter q was set to 5 and s to $40 + 2k$ where k is the number of medoids. For $CLARANS$

algorithm, in (Ng & Han 2002) indicates to keep a good balance between runtime and quality, the value of maxneighbor between 1.25 percent and 1.5 percent is very reasonable. Therefore, the parameters *numlocal* and *maxneighbor* were set to 5 and 750 ($kn \times 1.25\%$, where n is the number of objects), respectively. As shown in Table 2.6, $CLARANS\text{-}MITP$, $CLARANS\text{-}M$ and $CLARANS\text{-}ITP$ performed better than $CLARA$ both in the distance calculations and the average distance per object. Comparing with $CLARANS$, $CLARANS\text{-}MITP$, $CLARANS\text{-}M$ and $CLARANS\text{-}ITP$ can reduce the number of distance computations by 91%, 89% and 51%, respectively. Similarly, there is a significant reduction in execution time.

2.5.2 Gauss-Markov Source

A dataset of 3,000 objects in 8 dimensions were generated from a Gauss-Markov source (of the form $y_n = \alpha y_{n-1} + w_n$) where w_n is a zero-mean, unit variance, Gaussian white noise process, with $\alpha = 0.5$. For this experiment, 32 medoids were selected from 3,000 objects. For $CLARA$, q was set to 5 and s was set to $320 + 2k$, where k is the number of medoids. For $CLARANS$, the parameters *numlocal* and *maxneighbor* were set to 5 and 1200, respectively. Experimental results are shown in Table 2.7 and Figure 2.19, and show that compared with $CLARANS$, $CLARANS\text{-}MITP$, $CLARANS\text{-}M$ and $CLARANS\text{-}ITP$ can reduce the number of distance computations by 95%, 93% and 67%, respectively. For the measurement of executive time, there is a significant reduction in the execution time.

For $CLASA$, the parameters for the final temperature T_f, the initial temperature T_0, η, the number of total distance drop *disdrop*, *per* and the total number of perturbations *totalper* were set to 0.000025, 10, 0.95, 10, 200, 500000, respectively. As shown in Table 2.7 and Table 2.8, with these settings $CLASA$ performed better than $CLARA$ both in the computation time and the aver-

age distance per object. For the similar average distance, $CLASA$ reduced the number of distance calculations for more than 38% compared with $CLARANS$. The experiments are also shown a satisfactory results in these three extended $CLASA$ algorithms, comparing with $CLARANS$, $CLASA$-$MITP$, $CLASA$-M and $CLASA$-ITP may reduce the number of distance computations by 96%, 96% and 79%, respectively. The table also shows a significant reduction in the execution time. Obviously, the experimental results confirm the usefulness of these proposed search approaches.

2.5.3 Rectangular Clusters

A dataset of 3,000 objects collected from 20 rectangular clusters and 150 objects with two dimensions belonged to each clusters. For $CLARA$, the parameter q was set to 5 and s to $200+2k$. For the $CLARANS$, the parameter *numlocal* was set to 5 and parameter *maxneighbor* was set to 750. Experimental results are shown in Table 2.9, with these settings $CLARANS$-$MITP$, $CLARANS$-M and $CLARANS$-ITP performed better than $CLARA$ both in the computation time and the average distance per object. Based on the similar average distance per object, comparing with $CLARANS$, $CLARANS$-$MITP$, $CLARANS$-M and $CLARANS$-ITP can reduced the number of distance calculations by 93%, 90% and 89%, respectively.

For $CLASA$, the parameters for the final temperature T_f, the initial temperature T_0, η, the number of total distance drop *disdrop*, *per* and the total number of perturbations *totalper* were set to 0.000025, 10, 0.95, 10, 200, 50000, respectively. As shown in Table 2.9 and Table 2.10, with these settings $CLASA$ performed better than $CLARANS$ both in the computation time and the average distance per object. For the similar average distance, $CLASA$ reduced the number of distance calculations for more than 11% compared with $CLARANS$. The experiments are also shown a satisfactory results in these three extended

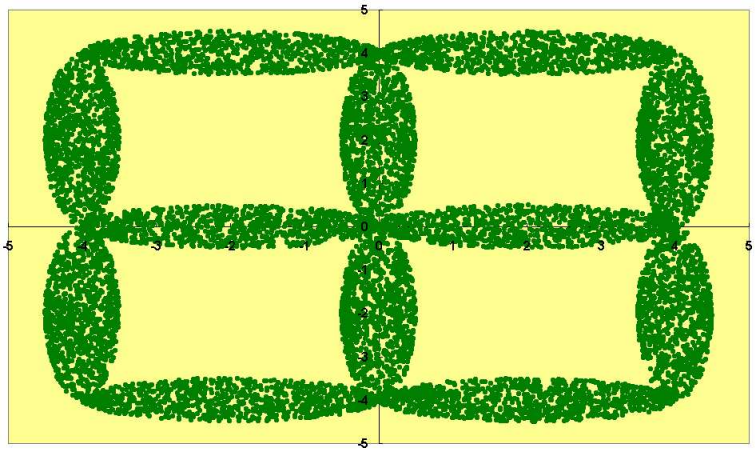


Figure 2.17. Twelve Elliptic Clusters

$CLASA$ algorithms, comparing with $CLARANS$, $CLASA$-$MITP$, $CLASA$-M and $CLASA$-ITP may reduce the number of distance computations by 98%, 95% and 98%, respectively. Again, there is a significant reduction in the execution time.

2.5.4 Elliptic Clusters

A dataset of 12,000 objects in 2 dimensions collected from twelve elliptic clusters were used for experiment as shown in Figure 2.17. 12 medoids were selected from 12,000 objects. For $CLARA$, the parameter q was set to 5 and s was set to $960 + 2k$. For $CLARANS$, the parameters *numlocal* and *maxneighbor* were set to 5 and 1,800, respectively. It can be seen in Table 2.11 and Figure 2.20 that, compared with $CLARANS$, $CLARANS$-$MITP$, $CLARANS$-M and $CLARANS$-ITP can reduce the number of distance computations by 88%, 84% and 84%, respectively. The result also shows a significant reduction in the execution time.

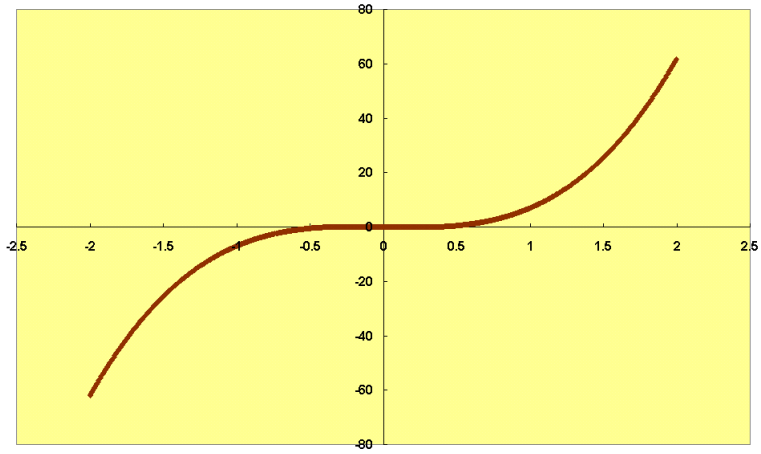


Figure 2.18. Curved Clusters

2.5.5 Curved Clusters

A dataset of 5,000 objects in 2 dimensions was generated from curve datasets as shown in Figure 2.18. The object (x, y) is collected from the area $-2 \leq x \leq 2$ and $y = 8x^3 - x$. 20 medoids were selected from 5,000 objects. For $CLARA$, q was set to 5 and s to $400 + 2k$. For $CLARANS$, the parameters *numlocal* and *maxneighbor* were set to 5 and 1,250, respectively. Results are shown in Table 2.12 and Figure 2.21 that, compared with $CLARANS$, $CLARANS\text{-}MITP$, $CLARANS\text{-}M$ and $CLARANS\text{-}ITP$ we can reduce the number of distance computations by 93%, 90% and 91%, respectively.

2.5.6 *Lena* Dataset

The *Lena* gray-level image data with size 512 by 512 is used for the sixth experiment. 16,384 objects with 16 dimensions were extracted from this image. 8 medoids were selected from these 16,384 objects. For $CLARA$, the parameter q was set to 5 and s was set to $1,000 + 2k$. For $CLARANS$, the parameters *numlocal* and *maxneighbor* were set to 5 and 1,800, respectively. In order to facilitate and enrich the effectiveness and efficiency, we decide to deepen our undestanding of these proposed search strategies–16, 32, 64 medoids were selected

from these 16,384 objects, respectively. Table 2.13 and Table 2.14 present traces of a typical run of these algorithms for *Lena* dataset. This can be observed that, in which the proposed $CLARANS\text{-}MITP$, $CLARANS\text{-}M$ and $CLARANS\text{-}ITP$ are superior to the $CLARA$ and $CLARANS$.

Due to $CLARA$ is computationally expensive in a large dataset with more medoids; we omit to calculate the computation complexity of $CLARA$ in these experiments which were selected 32 and 64 medoids for the purpose of comparison. With same parameters settings we observed that, a much better performance is obtained in the experiment that 64 medoids were selected for *Lena* dataset. Comparing with $CLARANS$, the Table 2.15 indicates that the proposed $CLARANS\text{-}MITP$, $CLARANS\text{-}M$ and $CLARANS\text{-}ITP$ can reduce the number of distance computations by 98%, 96% and 94%, respectively.

2.5.7 Real-World Dataset

In this experiment, the co-occurrence texture, one of image features extracted from the Corel image collection [1] is used for the final experiment. There are 68,040 photo images mostly of natural scenes from various categories. These images are converted to 16 dimensions gray-scale images. 10 medoids were selected from these 68,040 objects. For $CLARANS$, the parameters *numlocal* and *maxneighbor* were set to 5 and 8,505, respectively. Experimental results are shown in Table 2.16, with these settings $CLARANS\text{-}MITP$, $CLARANS\text{-}M$ and $CLARANS\text{-}ITP$ outperformed better than $CLARANS$ with the same average distance per object. Comparing with $CLARANS$, $CLARANS\text{-}MITP$, $CLARANS\text{-}M$ and $CLARANS\text{-}ITP$ can reduced the number of distance calculations by 87%, 80% and 77%, respectively. The table also shows a significant reduction in the execution time.

[1] The Corel collection of images available online at
`http://kdd.ics.uci.edu/databases/CorelFeatures/CorelFeatures.html`.

2.5.8 Summary

In our work, various extended versions of $CLARANS$ are presented based on enhanced search strategies. In addition, a new algorithm based on simulated annealing was presented and it too was amended with these search strategies.

Experimental results demonstrate that applying a hybrid search method using previous medoid index, utilization of memory, the criterion of TIE and partial distance search to $CLARANS$ can reduce the number of distance computations from 87% to 98%. In terms of the running time, the proposed $CLARANS$-$MITP$ algorithm can reduce the computation time up to 96%. Experiments with $CLASA$ also indicate an improvement over $CLARANS$ which is improved still further using the search strategies discussed. AS shown in Figure 2.22 and 2.23, they show that the extensions suggested can provide significant improvement, of in some cases up to 98%, over $CLARA$ and $CLARANS$. While other, more efficient algorithms have been developed, few show such improvements.Moreover, in many cases the techniques outlined here will be applicable.

It is important to note that the proposed search strategies can also be applied to other clustering algorithms. Indeed, as part of our future work, we aim extend the ideas in this chapter to other well-known clustering algorithms.

Table 2.6. Results of Experiment for Gaussian Source

Seed	*CLARA*			*CLARANS*		*CLARANS-ITP*		*CLARANS-M*		*CLARANS-MITP*		
	Distance Calcs (10^5)	Average Distance	Running Time (secs)	Distance Calcs (10^5)	Running Time (secs)	Distance Calcs (10^5)	Running Time (secs)	Distance Calcs (10^5)	Running Time (secs)	Distance Calcs (10^5)	Running Time (secs)	Average Distance
1	851	4.748	16	205	4	99	3	21	1	18	1	4.504
2	779	4.720	15	247	5	119	4	35	1	21	1	4.479
3	880	4.595	17	201	4	98	3	21	1	17	1	4.475
4	807	4.698	15	200	4	97	3	20	1	17	1	4.467
5	721	4.714	14	249	5	121	4	25	1	22	1	4.524
6	836	4.753	16	194	4	93	3	19	1	17	1	4.550
7	908	4.829	17	289	6	141	5	29	1	25	1	4.461
8	908	4.781	17	231	5	110	4	23	1	20	1	4.540
9	836	4.788	16	295	6	143	5	30	1	25	1	4.489
10	880	4.732	17	265	5	128	4	27	1	23	1	4.528
Ave.	841	4.736	16	238	5	115	4	24	1	20	1	4.502

Table 2.7. Results of Experiment for Gauss-Markov Source

Seed	*CLARA*			*CLARANS*		*CLARANS-ITP*		*CLARANS-M*		*CLARANS-MITP*		Average Distance
	Distance Calcs (10^7)	Average Distance	Running Time (secs)	Distance Calcs (10^7)	Running Time (secs)	Distance Calcs (10^7)	Running Time (secs)	Distance Calcs (10^7)	Running Time (secs)	Distance Calcs (10^7)	Running Time (secs)	
1	1548	4.559	2790	376	662	126	410	23	47	18	49	4.432
2	1637	4.592	2942	532	934	177	576	33	64	25	62	4.359
3	1789	4.551	3218	375	659	123	406	23	47	18	44	4.381
4	1726	4.578	3111	406	713	133	437	25	49	19	47	4.398
5	1827	4.559	3289	397	697	131	428	25	49	19	46	4.367
6	1675	4.526	3012	414	727	137	448	26	50	20	48	4.384
7	1853	4.527	3331	323	568	106	348	20	44	15	38	4.380
8	1624	4.483	2920	396	697	130	425	25	47	19	46	4.394
9	1903	4.545	3426	367	646	120	394	23	45	17	45	4.377
10	1802	4.514	3245	358	634	119	388	22	44	17	41	4.406
Ave.	1738	4.543	3128	394	694	130	426	24	49	19	47	4.388

Table 2.8: Results of Experiment for Gauss-Markov Source–Compared with *CLARANS* and Extended *CLASA* Algorithms

Seed	*CLARANS*			*CLASA*		*CLASA-ITP*		*CLASA-M*		*CLASA-MITP*		Average Distance
	Distance Calcs (10^7)	Average Distance	Running Time (secs)	Distance Calcs (10^7)	Running Time (secs)	Distance Calcs (10^7)	Running Time (secs)	Distance Calcs (10^7)	Running Time (secs)	Distance Calcs (10^7)	Running Time (secs)	
1	376	4.432	662	243	479	79	308	15	37	12	31	4.366
2	532	4.359	934	242	479	80	311	15	39	12	35	4.362
3	375	4.381	659	241	475	79	309	15	37	11	30	4.334
4	406	4.398	713	242	479	80	309	15	39	12	31	4.344
5	397	4.367	697	242	477	80	310	15	37	12	31	4.343
6	414	4.384	727	241	478	79	307	15	39	11	35	4.374
7	323	4.380	568	245	483	80	312	15	37	12	31	4.401
8	396	4.394	697	241	478	79	307	15	38	11	30	4.364
9	367	4.377	646	242	478	80	309	15	37	12	31	4.356
10	358	4.406	634	242	478	80	309	15	39	11	31	4.377
Ave.	394	4.388	694	242	478	80	309	15	38	12	32	4.362

Table 2.9. Results of Experiment for Rectangular Clusters

Seed	*CLARA*			*CLARANS*		*CLARANS-ITP*		*CLARANS-M*		*CLARANS-MITP*		Average Distance
	Distance Calcs (10^7)	Average Distance	Running Time (secs)	Distance Calcs (10^7)	Running Time (secs)	Distance Calcs (10^7)	Running Time (secs)	Distance Calcs (10^7)	Running Time (secs)	Distance Calcs (10^7)	Running Time (secs)	
1	192	0.181	110	167	97	18	36	16	12	11	11	0.172
2	205	0.185	120	132	76	14	29	13	9	9	10	0.177
3	169	0.182	97	120	70	13	26	12	9	8	8	0.169
4	219	0.186	127	138	80	15	31	14	10	9	10	0.173
5	174	0.184	101	178	103	19	39	17	12	12	12	0.172
6	205	0.181	118	144	84	15	32	14	10	10	10	0.174
7	172	0.183	100	155	90	17	34	15	11	10	10	0.176
8	205	0.183	119	157	91	17	35	15	11	10	11	0.175
9	207	0.185	120	147	85	16	32	14	10	10	10	0.176
10	180	0.182	104	177	103	19	39	17	12	12	12	0.175
Ave.	193	0.183	112	151	88	16	33	15	11	10	10	0.174

Table 2.10: Results of Experiment for Rectangular Clusters–Compared with *CLARANS* and Extended *CLASA* Algorithms

Seed	*CLARANS*			*CLASA*		*CLASA-ITP*		*CLASA-M*		*CLASA-MITP*		Average Distance
	Distance Calcs (10^7)	Average Distance	Running Time (secs)	Distance Calcs (10^7)	Running Time (secs)	Distance Calcs (10^7)	Running Time (secs)	Distance Calcs (10^7)	Running Time (secs)	Distance Calcs (10^7)	Running Time (secs)	
1	167	0.172	97	134	88	2	34	7	10	2	10	0.174
2	132	0.177	76	136	88	2	35	8	10	2	10	0.171
3	120	0.169	70	136	89	2	35	8	11	2	10	0.173
4	138	0.173	80	134	87	2	35	7	10	2	10	0.173
5	178	0.172	103	132	86	2	34	7	10	2	10	0.175
6	144	0.174	84	133	87	2	35	7	10	2	10	0.171
7	155	0.176	90	132	86	2	34	7	10	2	10	0.173
8	157	0.175	91	136	89	2	36	8	10	2	10	0.172
9	147	0.176	85	135	88	2	35	7	10	2	10	0.173
10	177	0.175	103	136	88	2	36	8	10	2	10	0.173
Ave.	151	0.174	88	134	87	2	35	7	10	2	10	0.173

Table 2.11. Results of Experiment for Elliptic Clusters

Seed	*CLARA*			*CLARANS*		*CLARANS-ITP*		*CLARANS-M*		*CLARANS-MITP*		Average Distance
	Distance Calcs (10^7)	Average Distance	Running Time (secs)	Distance Calcs (10^7)	Running Time (secs)	Distance Calcs (10^7)	Running Time (secs)	Distance Calcs (10^7)	Running Time (secs)	Distance Calcs (10^7)	Running Time (secs)	
1	1157	0.940	684	918	715	143	370	147	298	105	280	0.931
2	925	0.942	547	635	483	100	254	102	202	74	203	0.943
3	1225	0.956	721	769	584	120	305	123	260	89	235	0.935
4	1034	0.951	612	796	603	125	322	127	265	92	253	0.928
5	1116	0.946	659	900	686	142	359	144	287	104	273	0.949
6	1089	0.960	647	721	548	114	291	115	240	84	218	0.936
7	776	0.972	455	849	635	132	337	136	278	98	264	0.936
8	939	0.944	556	931	714	146	369	149	301	108	282	0.934
9	844	0.954	493	592	449	94	241	95	189	69	191	0.930
10	844	0.942	489	751	576	118	298	120	236	87	234	0.931
Ave.	995	0.951	586	786	599	123	315	126	256	91	243	0.935

Table 2.12. Results of Experiment for Curved Clusters

Seed	*CLARA*			*CLARANS*		*CLARANS-ITP*		*CLARANS-M*		*CLARANS-MITP*		Average Distance
	Distance Calcs (10^7)	Average Distance	Running Time (secs)	Distance Calcs (10^7)	Running Time (secs)	Distance Calcs (10^7)	Running Time (secs)	Distance Calcs (10^7)	Running Time (secs)	Distance Calcs (10^7)	Running Time (secs)	
1	741	2.232	425	652	397	53	148	64	86	40	58	2.151
2	734	2.278	432	626	372	51	123	61	71	38	50	2.181
3	847	2.317	498	702	422	57	151	69	80	43	67	2.179
4	797	2.342	463	719	425	58	141	70	82	44	67	2.192
5	840	2.321	475	574	341	45	122	56	65	35	46	2.156
6	805	2.252	465	623	368	49	121	61	59	38	57	2.157
7	741	2.305	432	650	398	52	139	64	74	39	60	2.171
8	797	2.238	463	636	378	51	134	62	60	39	59	2.144
9	741	2.209	429	551	339	44	107	54	63	34	54	2.185
10	805	2.317	473	484	285	38	102	47	46	29	43	2.165
Ave.	785	2.281	455	622	373	50	129	61	69	38	56	2.168

Table 2.13: Experimental Results for Ten Runs of *CLARANS*, *CLARANS-ITP*, *CLARANS-M* and *CLARANS-MITP* Algorithms for 8 Medoids of *Lena* Dataset

Seed	*CLARA*			*CLARANS*		*CLARANS-ITP*		*CLARANS-M*		*CLARANS-MITP*		
	Distance Calcs (10^7)	Average Distance	Running Time (secs)	Distance Calcs (10^7)	Running Time (secs)	Distance Calcs (10^7)	Running Time (secs)	Distance Calcs (10^7)	Running Time (secs)	Distance Calcs (10^7)	Running Time (secs)	Average Distance
1	299	2829.559	1014	797	2985	183	1197	187	1108	130	949	2816.426
2	312	2833.360	1057	568	2127	130	853	133	795	92	678	2831.768
3	325	2839.221	1100	761	2853	174	1133	179	1066	123	904	2806.479
4	358	2821.522	1216	597	2236	137	896	140	835	97	710	2814.164
5	286	2827.433	969	442	1656	102	664	104	619	72	530	2821.410
6	286	2842.224	972	544	2034	123	809	127	750	87	643	2822.102
7	306	2838.232	1257	559	2094	128	835	131	776	90	666	2817.135
8	286	2843.913	1179	568	2128	131	856	133	789	93	680	2819.383
9	247	2820.759	1210	646	2419	148	968	151	905	104	769	2823.581
10	286	2819.524	966	590	2210	135	884	138	819	96	701	2826.953
Ave.	299	2831.575	1094	607	2274	139	910	142	846	98	723	2819.940

Table 2.14: Experimental Results for Ten Runs of *CLARANS*, *CLARANS-ITP*, *CLARANS-M* and *CLARANS-MITP* Algorithms for 16 Medoids of *Lena* Dataset

Seed	*CLARA*			*CLARANS*		*CLARANS-ITP*		*CLARANS-M*		*CLARANS-MITP*		
	Distance Calcs (10^7)	Average Distance	Running Time (secs)	Distance Calcs (10^7)	Running Time (secs)	Distance Calcs (10^7)	Running Time (secs)	Distance Calcs (10^7)	Running Time (secs)	Distance Calcs (10^7)	Running Time (secs)	Average Distance
1	2220	2108.098	8328	1249	4984	164	1152	152	887	97	719	2064.426
2	2273	2093.875	8541	1530	6110	203	1424	186	1076	120	899	2082.017
3	2114	2115.623	7943	1417	5651	183	1303	172	1007	108	813	2068.945
4	2379	2070.973	7936	1216	4850	157	1116	148	857	93	703	2086.675
5	2114	2158.374	7033	1756	7010	228	1617	213	1247	135	1010	2065.860
6	2643	2071.464	8791	1313	5204	171	1213	159	924	101	764	2083.383
7	2167	2140.907	7224	1736	6884	226	1600	210	1230	134	999	2059.370
8	2009	2097.309	6742	1477	5683	192	1364	179	1045	113	855	2073.808
9	1982	2126.558	6633	2018	8009	263	1854	245	1425	157	1168	2064.848
10	1824	2089.289	6832	1312	4632	171	1205	159	920	102	763	2074.241
Ave.	2172	2107.247	7600	1502	5920	196	1385	182	1062	116	869	2072.357

Table 2.15. Results of Experiment for *Lena* Dataset

Medoids	*CLARANS*		*CLARANS-ITP*		*CLARANS-M*		*CLARANS-MITP*		Average Distance
	Distance Calcs (10^7)	Running Time (secs)	Distance Calcs (10^7)	Running Time (secs)	Distance Calcs (10^7)	Running Time (secs)	Distance Calcs (10^7)	Running Time (secs)	
64	10577	35458	530	4480	329	1888	187	1444	1257.929
32	4133	14181	326	2482	255	1469	153	1164	1606.656
16	1502	5920	196	1385	182	1062	116	869	2072.357
8	607	2274	139	910	142	846	98	723	2819.940

Table 2.16. Results of Experiment for Co-Occurence Texture of Corel Image Collection

Seed	*CLARANS*		*CLARANS-ITP*		*CLARANS-M*		*CLARANS-MITP*		Average Distance
	Distance Calcs (10^7)	Running Time (secs)	Distance Calcs (10^7)	Running Time (secs)	Distance Calcs (10^7)	Running Time (secs)	Distance Calcs (10^7)	Running Time (secs)	
1	12557	43754	2793	16473	2383	14390	1615	14001	4.600
2	18523	64562	4072	23900	3519	21286	2365	16934	4.612
3	13845	48110	3088	18187	2632	15536	1788	12742	4.632
4	19136	66388	4287	25336	3637	21151	2474	18505	4.610
5	15938	55768	3563	20953	3028	17781	2059	16435	4.648
6	15844	55024	3519	20498	3459	18123	2032	14147	4.646
7	15097	52272	3391	26678	2869	19456	1958	13966	4.618
8	11963	41403	2640	20762	2275	13516	1540	11037	4.618
9	13528	46781	2981	17519	2572	15222	1736	12264	4.619
10	18400	63728	4070	23739	3497	20476	2365	16884	4.571
Ave.	15483	53779	3440	21404	2987	17694	1993	14691	4.617

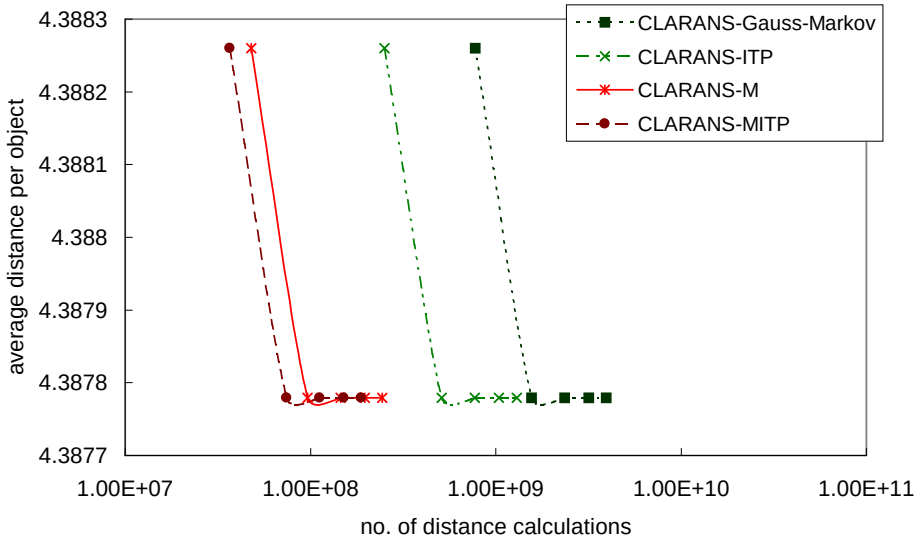


Figure 2.19. Results of Gauss-Markov Experiment

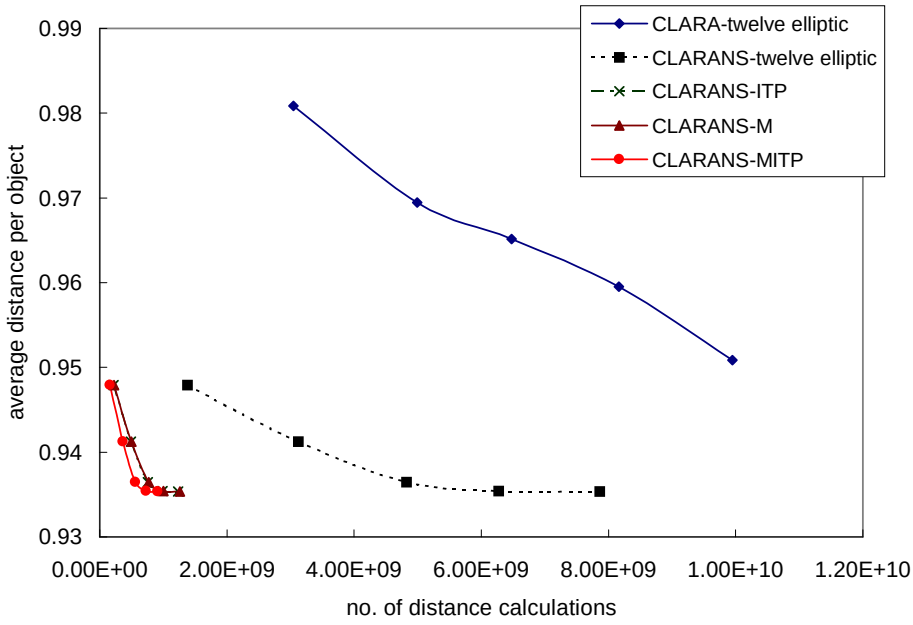


Figure 2.20. Results of Elliptic Clusters Experiment

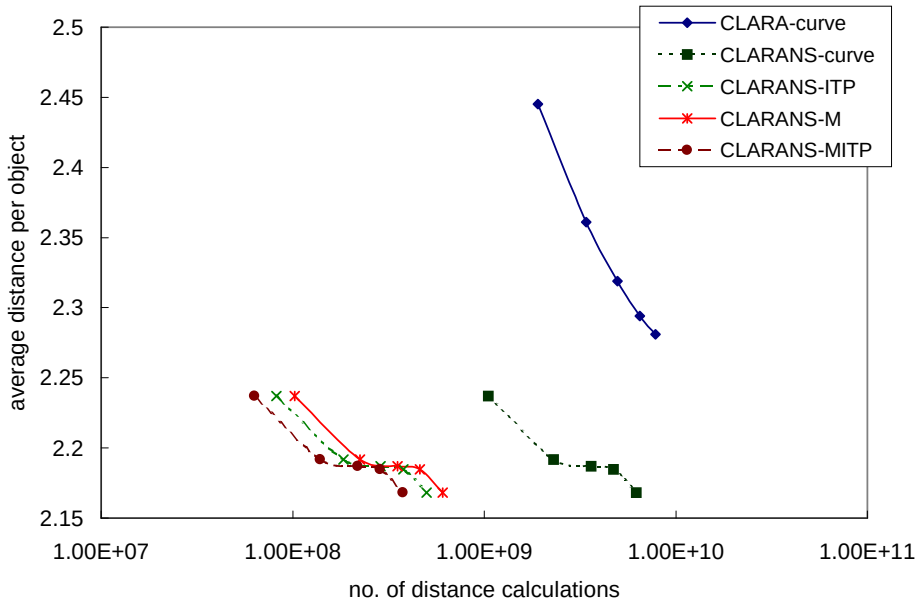


Figure 2.21. Results of Curved Clusters Experiment

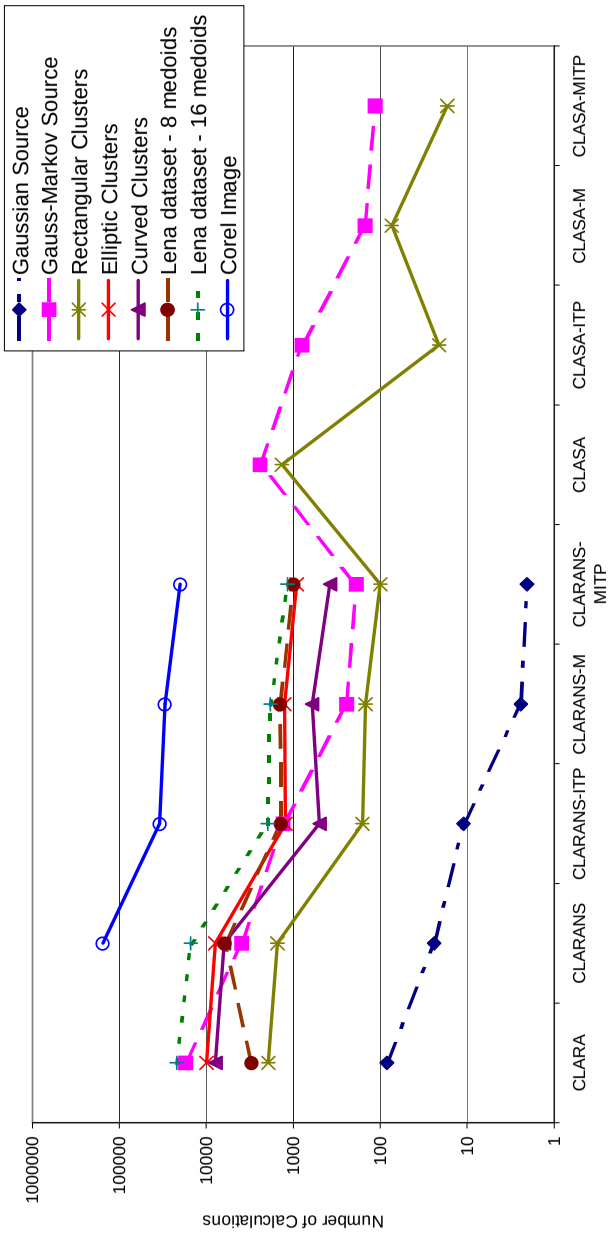


Figure 2.22. Results of Experiments by Number of Calculations

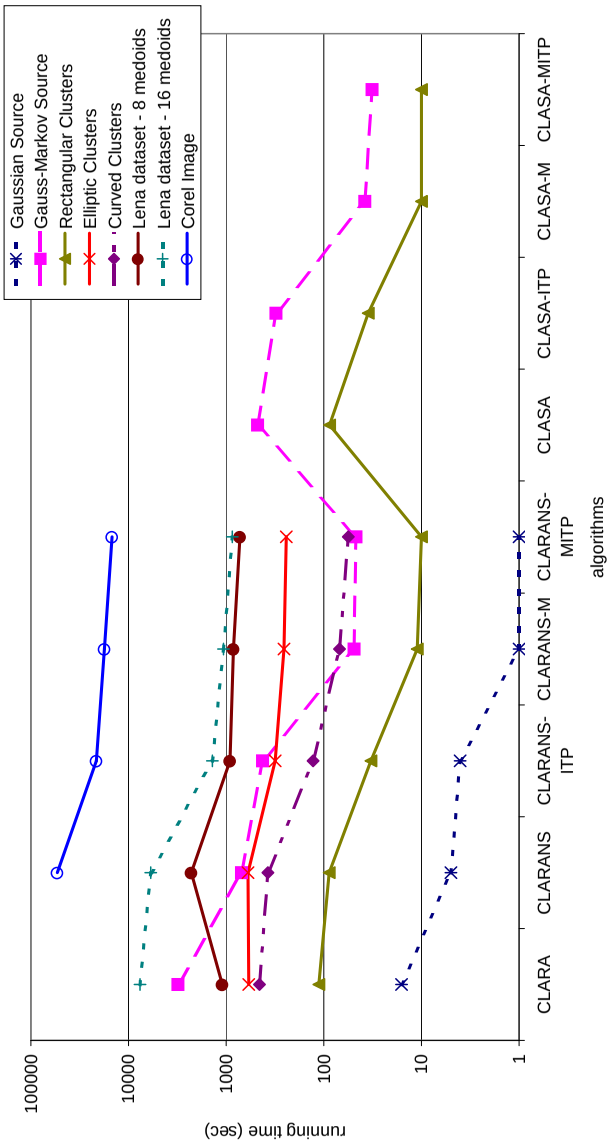


Figure 2.23. Results of Experiments by Number of Calculations

Chapter 3

Sampling Schemes for K-Medoids Algorithm

Clustering in data mining is used to group homogeneous objects based on the value of their attributes, connectivity, relative density, or some specific characteristics. The k-medoids-based algorithms have been shown to be effective to spherical-shaped clusters with outliers. However, they are not efficient for large database. In this chapter, we propose a novel *Multi-Centroids with Multi-Runs Sampling Scheme* ($MCMRS$) to improve the performance of many k-medoids-based algorithms, including PAM, $CLARA$ and $CLARANS$ (Chu, Roddick & Pan 2002a). $MCMRS$ is also further improved by combining with the $CLASA$ algorithm. In addition, we have also applied the adaptive concept to further improve this novel sampling scheme ($MCMRS$), *Incremental Multi-Centroid, Multi-Run Sampling Scheme* termed $IMCMRS$ by adjusting the $NumSample$, $NumRun$ and $NumCandidate$ automatically (Chu, Roddick & Pan 2002c, Chu, Roddick & Pan 2002d, Chu, Roddick & Pan 2004c). Experimental results demonstrate the proposed scheme can not only reduce by more than 80% computation time but also reduce the average distance per object compared with $CLARA$ and $CLARANS$. $IMCMRS$ is also superior to $MCMRS$.

3.1 Introduction

Clustering techniques have been studied extensively in many research area. There exist a large number of clustering algorithms in the literature including k-means (MacQueen 1967), k-medoids (Kaufman & Rousseeuw 1990), $CACTUS$ (Ganti, Gehrke & Ramakrishnan 1999), $CURE$ (Guha et al. 1998), $CHAMELEON$ (Karypis et al. 1999) and $DBSCAN$ (Ester et al. 1996). For k-means, each cluster is represented by the mean value of the objects in the cluster whereas in the k-medoids algorithm each cluster is represented by one of the objects located near the centre of the cluster. k-means can be sensitive to outliers (or noise) while k-medoids is generally more robust. The drawback of the k-medoids algorithms is the time complexity of determining the medoids. In order to ameliorate this shortcoming, a novel sampling scheme based on *Multi-Centroids with Multi-Runs Sampling scheme* ($MCMRS$) is proposed to improve k-medoids-based algorithms. Significantly, all existing k-medoids algorithms, including $CLASA$, can be improved by combining with $MCMRS$. Both the $MCMRS$ and the combined $MCMRS$-$CLASA$ are shown to be superior to PAM, $CLARA$, $CLARANS$ and $CLASA$. The proposed Multi-Centroids with Multi-Runs Sampling scheme ($MCMRS$) and the combined version, $MCMRS$-$CLASA$ algorithm are described in section 3.2.1. Although $MCMRS$ (Chu, Roddick & Pan 2002*a*, Chu, Roddick & Pan 2004*c*) improves the efficiency and effectiveness of the k-medoids algorithms, it can be further improved by adopting the adaptive concept. $IMCMRS$ not only can reduce the average distance but also speed the clustering process. The computation load in $IMCMRS$ can be further released by applying a more efficient centroid-based clustering method (Huang, Pan, Lu, Sun & Hang 2001). Section 3.2.2 and 3.3.2 will present some experimental results using three artificial databases and a real image database.

3.2 Multi-Centroid, Multi-Run Sampling Scheme ($MCMRS$)

3.2.1 Motivation

K-means and k-medoids are both popular partitioning clustering algorithms. K-means can be sensitive while k-medoids is generally more robust to outliers (or noise). One of the main factors to limit the use of the k-medoids algorithm is the inefficiency of k-medoids algorithms comparing with k-means – k-means algorithm can be several orders of magnitude faster than the k-medoids algorithm. In general, it is not efficient for k-medoids algorithm even for moderate sized datasets. This drawback can be overcome with the aid of an efficient sampling scheme. The idea for the sampling scheme in this section is motivated from the efficiency of the centroid-based clustering algorithms (Huang, Pan, Lu, Sun & Hang 2001, Huang, Chu, Pan & Lu 2001). From our empirical observations, we noticed that there is a higher probability of better medoids being selected within some distance from the centroids of the clusters. Based on this observation and the efficiency of the centroid-based clustering, we can generate k groups of medoid candidates with each group containing $NumCandidate$ nearest objects from the centroid for each centroid-based cluster. K medoids can be collected from each object in each group randomly. This process interates $NumSample$ times. This sampling scheme can be more robust by repeating the above procedure many times. The proposed $MCMRS$ can be depicted as follows:

Step 1:Repeat the following steps for $NumRun$ times.

Step 2:Get representative centroids by calling the centroid-based clustering algorithm (such as k-means or GLA algorithm (Linde, Buzo & Gray 1980)) with random initialization.

Step 3:Get $NumCandidate$ objects for each cluster by selecting the $NumCandidate$ nearest objects from the centroid in each cluster.

Step 4:Repeat the following steps $NumSample$ times.

Step 5:Generate medoids by selecting one object from the $NumCandidate$ nearest objects in each cluster.

Step 6:Calculate the average distance per object and update the best medoids.

$MCMRS$ sampling scheme can be combined with PAM, $CLARA$, $CLARANS$ and $CLASA$ to further improve the performance. The combined version of $MCMRS$ and $CLASA$ for example, (referred here as $MCMRS$-$CLASA$) can be described as follows:

Step 1:Choose the best centroids whose average distance per object is a minimum by running the centroid-based clustering $NumRun$ times.

Step 2:Get $NumCandidate$ objects for each cluster by selecting the $NumCandidate$ nearest objects from the centroid in each cluster.

Step 3:Call $CLASA$ by using the nearest object from the centroid in each cluster as the initial medoids. The candidate medoid is swapped from the $NumCandidate$ nearest objects in the same group.

Step 4:Terminate the program and return the medoids when the predefined criterion is satisfied.

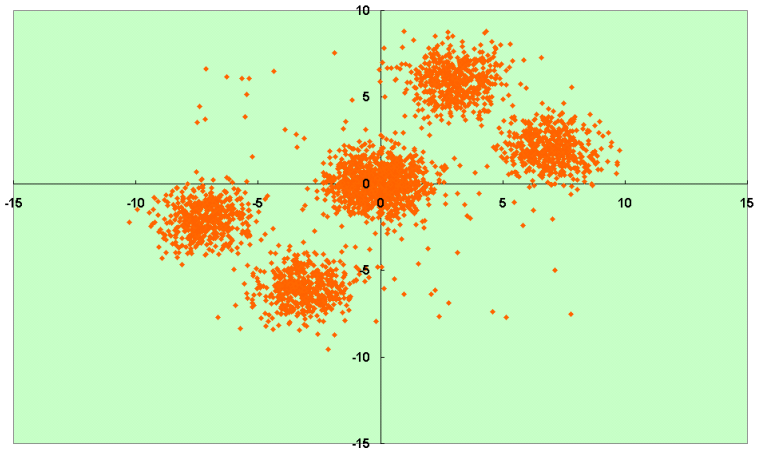


Figure 3.1. Compact Clusters with Noise

3.2.2 Experimental Results

Three artificial databases and one real image database were used for the experiments are as follows:

1. 1,500 objects collected from four elliptic clusters shown in Figure 2.5.
2. 12,000 objects collected from twelve elliptic clusters shown in Figure 2.17.
3. 3,100 objects collected from five compact clusters shown in Figure 3.1.
4. 16,384 objects with 16 dimensions are generated from the *Lena* grey-level image with size 512 by 512 .

Experiments were carried out to test the number of distance calculations and the average distance per object for the $CLARA$, $CLARANS$, $CLASA$ algorithms and the proposed $MCMRS$ and $MCMRS$-$CLASA$ algorithms. Squared Euclidean distance measure is used. The four elliptic clusters were used for the first experiment and 12 medoids are selected from 1500 objects. For $CLARA$, the parameter q was set to 5 and s was set to $160 + 2 \times k$.

For $CLARANS$, the parameter *numlocal* was set to 5 and parameter *maxneighbour* was set to 270 (i.e. 1.5% of $k \times T$). For $CLASA$, the parameters for the final temperature T_f, the initial temperature T_0, η, the times of distance drops *dis_drop*,

per and the total number of perturbations *total_per* were set to 0.000025, 0.0001, 0.95, 10, 200, 50000, respectively. For $MCMRS$, a k-means algorithm is used to generate 12 centroid-based clusters. The parameters $NumRun$, $NumCandidate$ and $NumSample$ in $MCMRS$ were set to 20, 10 and 200, respectively. For $MCMRS$-$CLASA$ algorithm, k-means algorithm is also used to generate 12 centroid-based clusters. The parameters $NumRun$ and $NumCandidate$ in $MCMRS$-$CLASA$ algorithm were set to 60 and 10, respectively. The other parameters in $MCMRS$-$CLASA$ were the same as for $CLASA$. The experimental results of $CLARA$, $CLARANS$, $MCMRS$ and $MCMRS$-$CLASA$ are averaged for 10 seeds are shown in Table 3.1 and the result of the first seed of $CLASA$ are shown in Figure 3.2. As shown in Figure 3.2, both $MCMRS$-$CLASA$ and $MCMRS$ algorithms outperform $CLASA$, $CLARANS$ and $CLARA$ algorithms.

Table 3.1. Results of Experiment for Four Elliptic Clusters

seed	*CLARA*		*CLARANS*		*MCMRS*		*MCMRS-CLASA*	
	Average dist.	Count of dist.(10^5)	Average dist.	Count of dist.(10^5)	Average dist.	Count of dist.(10^5)	Average dist.	Count of dist.(10^5)
1	0.228	2004	0.228	1609	0.216	808	0.212	541
2	0.221	2302	0.232	1067	0.217	821	0.213	567
3	0.236	2388	0.236	1351	0.218	825	0.212	527
4	0.230	2686	0.233	1154	0.218	811	0.213	572
5	0.227	2430	0.232	1572	0.216	807	0.213	546
6	0.237	2260	0.227	1223	0.217	803	0.213	549
7	0.232	2646	0.234	895	0.217	821	0.213	591
8	0.236	2516	0.231	1429	0.217	803	0.213	580
9	0.233	2345	0.231	1148	0.216	803	0.213	524
10	0.238	2728	0.231	1167	0.216	805	0.212	581
Ave.	0.232	2431	0.232	1140	0.218	811	0.213	558

Twelve elliptic clusters were used for the second experiment. 12 medoids are selected from 12,000 objects. For $CLARA$, the parameter q was set to 5 and s was set to $960 + 2 * k$. For $CLARANS$, the parameters *numlocal* and *maxneighbour* are set to 5 and 1800, respectively. For $MCMRS$, a k-means algorithm is used to generate 12 centroid-based clusters. The parameters $NumRun$, $NumCandidate$ and $NumSample$ in $MCMRS$ were set to 20, 10 and 200, re-

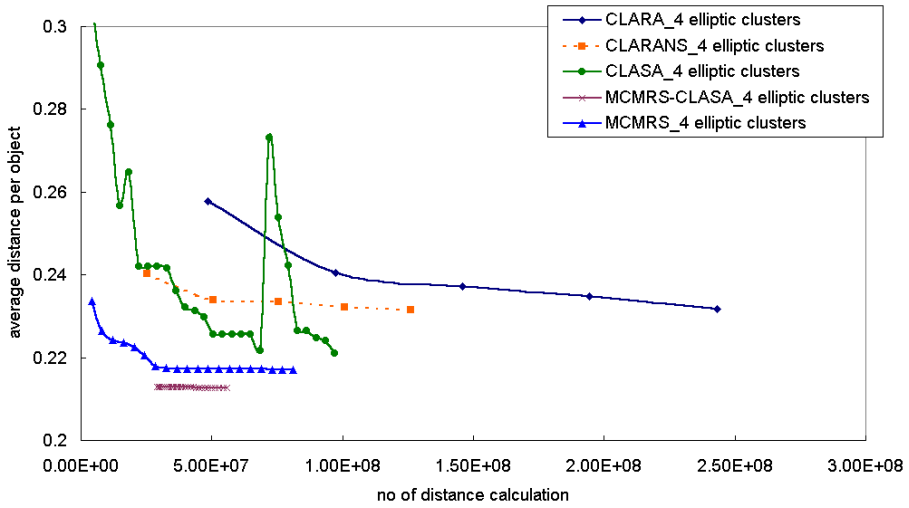


Figure 3.2: Performance comparison of *CLARA*, *CLARANS*, *CLASA*, *MCMRS-CLASA* and *MCMRS* for Four Elliptic Clusters

spectively. For *MCMRS-CLASA*, a k-means algorithm is also used to generate 12 centroid-based clusters. The parameters *NumRun* and *NumCandidate* in *MCMRS-CLASA* were set to 60 and 10, respectively. The parameters for the final temperature T_f, the initial temperature T_0, η, the number of total distance drop *dis_drop*, *per* and the total number of perturbations *total_per* were set to 0.000025, 0.001, 0.95, 5, 20, 500000, respectively. Experimental results for 10 runs for *CLARA*, *CLARANS*, *MCMRS* and *MCMRS-CLASA* are shown in Table 3.2 and Figure 3.3.

The compact clusters with noise were used for the third experiment. 5 medoids are selected from 3,100 objects. For *CLARA*, the parameter q was set to 5 and s was set to $200 + 2 * k$. For *CLARANS*, the parameters *numlocal* and *maxneighbour* are set to 5 and 200, respectively. For *MCMRS*, a k-means algorithm is used to generate 5 centroid-based clusters. The parameters *NumRun*, *NumCandidate* and *NumSample* in *MCMRS* were set to 20, 10 and 200, respectively. For *MCMRS-CLASA*, a k-means algorithm was again used to generate 5 centroid-based clusters. The parameters *NumRun* and *NumCandidate* in *MCMRS-CLASA* were set to 60 and 10, respectively. The parameters for the

Table 3.2. Results of Experiment for Twelve Elliptic Clusters

seed	*CLARA*		*CLARANS*		*MCMRS*		*MCMRS-CLASA*	
	Average dist.	Count of dist.(10^5)	Average dist.	Count of dist.(10^5)	Average dist.	Count of dist.(10^5)	Average dist.	Count of dist.(10^5)
1	0.940	115659	0.931	91792	0.922	6587	0.920	5217
2	0.942	92528	0.943	63512	0.920	6554	0.918	5479
3	0.956	122462	0.935	76873	0.930	6496	0.920	5312
4	0.951	103413	0.928	79582	0.928	6447	0.918	5373
5	0.946	111577	0.949	90014	0.921	6483	0.918	5374
6	0.960	108856	0.936	72069	0.922	6632	0.918	5407
7	0.972	77562	0.936	84914	0.920	6426	0.920	5509
8	0.944	93889	0.934	93092	0.920	6549	0.918	5413
9	0.954	84365	0.930	59242	0.923	6573	0.918	5426
10	0.942	84365	0.931	75053	0.923	6541	0.920	5171
Ave.	0.951	99467	0.935	78615	0.923	6529	0.919	5368

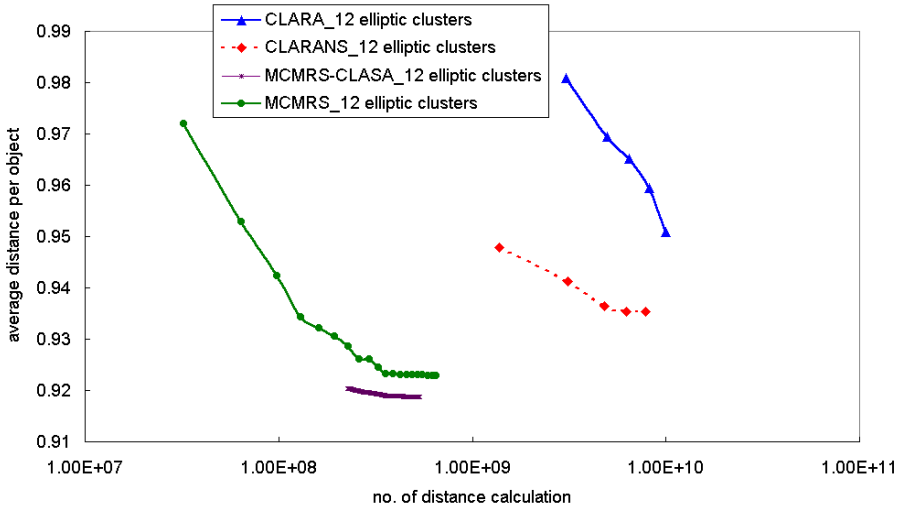


Figure 3.3: Performance Comparison of *CLARA*, *CLARANS*, *MCMRS* and *MCMRS-CLASA* for Twelve Elliptic Clusters

final temperature T_f, the initial temperature T_0, η, the times of distance drops *dis_drop*, *per* and the total number of perturbations *total_per* were set to 0.00025, 0.001, 0.85, 10, 200, 50000, respectively. Experimental results based on 10 runs for *CLARA*, *CLARANS*, *MCMRS* and *MCMRS-CLASA* are shown in Figure 3.4. If the database is not large and the medoid size is small, the performance of *CLARA* is better than *CLARANS* shown in Table 3.3 and Figure 3.4. Both

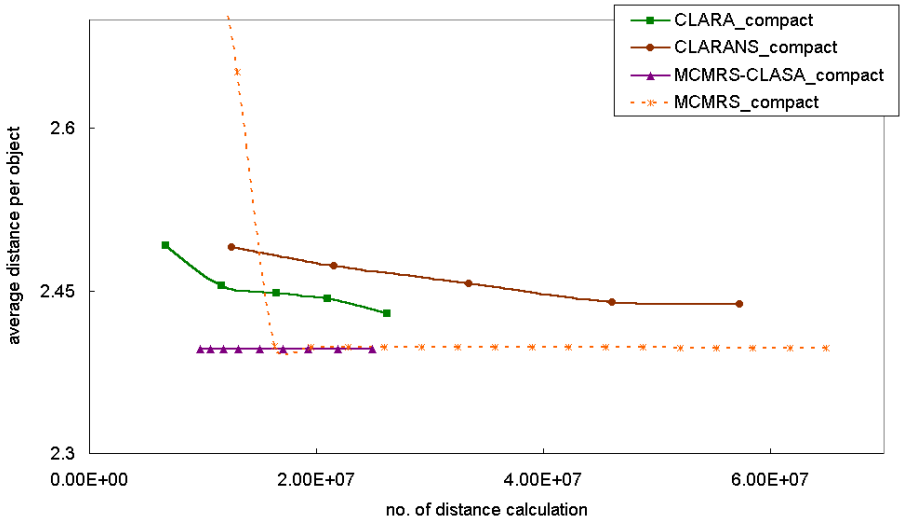


Figure 3.4: Performance Comparison of $CLARA$, $CLARANS$, $MCMRS$ and $MCMRS$-$CLASA$ for Compact Clusters with Noise

$MCMRS$ and $MCMRS$-$CLASA$ are more efficient and effective than $CLARA$ and $CLARANS$.

Table 3.3. Results of Experiment for Compact Clusters

seed	$CLARA$		$CLARANS$		$MCMRS$		$MCMRS$-$CLASA$	
	Average dist.	Count of dist.(10^5)	Average dist.	Count of dist.(10^5)	Average dist.	Count of dist.(10^5)	Average dist.	Count of dist.(10^5)
1	2.436	253	2.432	584	2.398	646	2.397	243
2	2.425	274	2.457	600	2.397	647	2.397	232
3	2.430	264	2.429	604	2.398	655	2.397	239
4	2.440	295	2.442	504	2.397	649	2.397	251
5	2.445	232	2.431	583	2.398	651	2.397	232
6	2.411	253	2.419	606	2.398	655	2.397	244
7	2.435	243	2.457	530	2.397	642	2.397	256
8	2.422	285	2.470	519	2.397	654	2.397	241
9	2.444	253	2.417	620	2.397	647	2.397	267
10	2.440	274	2.424	581	2.398	645	2.397	283
Ave.	2.429	263	2.438	573	2.398	649	2.397	249

The *Lena* gray-level image data with size 512 by 512 is used for the fourth experiment. 16,384 objects with 16 dimensions are extracted from this image. 8 medoids are selected from these 16,384 objects. For $CLARA$, the parameter q was set to 5 and s was set to $1000 + 2 * k$. For $CLARANS$, the parameters

numlocal and *maxneighbour* are set to 5 and 1800, respectively. For $MCMRS$, a k-means algorithm is used to generate 8 centroid-based clusters. The parameters $NumRun$, $NumCandidate$ and $NumSample$ in $MCMRS$ were set to 20, 10 and 200, respectively. For $MCMRS$-$CLASA$, a k-means algorithm is used to generate 8 centroid-based clusters. The parameters $NumRun$ and $NumCandidate$ in $MCMRS$-$CLASA$ were set to 20 and 10, respectively. The parameters for the final temperature T_f, the initial temperature T_0, η, the times of distance drops *dis_drop*, *per* and the total number of perturbations *total_per* were set to 0.000025, 0.0001, 0.85, 10, 200, 50000, respectively. Experimental results based on 10 runs for $CLARA$, $CLARANS$, $MCMRS$ and $MCMRS$-$CLASA$ are shown in Table 3.4. The proposed $MCMRS$-$CLASA$ algorithm can reduce the computation time by approximately a factor of 20 and obtain better average distance comparing with $CLARANS$.

Table 3.4. Results of Experiment for *Lena* Image

seed	*CLARA*		*CLARANS*		*MCMRS*		*MCMRS-CLASA*	
	Average dist.	Count of dist.(10^5)	Average dist.	Count of dist.(10^5)	Average dist.	Count of dist.(10^5)	Average dist.	Count of dist.(10^5)
1	2829.56	30057	2816.43	79704	2800.15	6121	2795.64	3135
2	2833.36	31379	2831.77	56796	2802.49	5897	2795.64	3135
3	2839.22	32700	2806.48	76182	2801.81	6138	2795.64	3135
4	2821.52	36003	2814.16	59684	2801.98	6080	2795.76	3264
5	2827.43	28736	2821.41	44216	2802.71	6067	2795.64	3287
6	2842.22	28736	2822.10	54274	2801.52	6158	2793.44	3184
7	2838.23	30718	2817.13	55947	2800.16	6145	2795.76	3192
8	2843.91	28736	2819.38	56845	2803.82	6036	2795.76	3179
9	2820.76	24772	2823.58	64621	2802.98	6229	2796.01	3387
10	2819.52	28736	2826.95	59039	2801.34	6019	2793.54	3065
Ave.	2831.57	30057	2819.94	60731	2801.90	6089	2795.28	3196

Experimental results based on three artificial databases and one real image database indicates that the proposed $MCMRS$ and $MCMRS$-$CLASA$ algorithms can not only reduce the average distance but also improve the clustering process. The computation load in $MCMRS$ and $MCMRS$-$CLASA$ can be further released by applying a more efficient centroid-based clustering method (Huang,

Pan, Lu, Sun & Hang 2001).

3.3 Incremental Multi-Centroid, Multi-Run Sampling Scheme (IMCMRS)

3.3.1 Motivation

Although $MCMRS$ improves the efficiency and effectiveness of the k-medoids algorithms, it can be further improved by adopting the adaptive concept. The idea of an *Incremental Multi-Centroid, Multi-Run Sampling Scheme* ($IMCMRS$) is based on the observation that better medoids are not always found near the centres of the clusters for each run of the centroid-based clustering algorithm. In order to reduce computation time, the sampling times and the number of candidate objects should be increased when obtaining better results with centroid-based clustering and conversely reduced when obtaining worse than average results.

Based on this observation and given the efficiency of the centroid-based clustering, we can generate $NumRun$ groups of medoid candidates with each group containing several objects close to the centroid for each centroid-based cluster. k medoids can be collected from each object in each group randomly. This process iterates $NumSample$ times. The groups with poor results are deleted and more objects are chosen from the better groups. As shown in Figure 3.5, the $IMCMRS$ sampling scheme can be described as follows:

1. Get $NumRun$ groups representative centroids by calling the centroid-based clustering algorithm (such as k-means or GLA (Linde et al. 1980)) with random initialisation for $NumRun$ times.

2. Get the nearest object to the centroid for each cluster and choose the nearest objects as the medoids. Calculate the average distance per object. Set $n = 1$, where n is the iteration count.

3. Set $NumRun = NumRun/2$ and choose $NumRun$ groups with better average distance and get $O = NumCandidate + NumSelect \times n$ objects for each cluster by sorting the O nearest objects from the centroid in each cluster.

4. Repeat the following $NumSample + StepSample \times n$ times.

   (a) Generate medoids by selecting one object from the nearest objects in each cluster.

   (b) Calculate the average distance per object and update the best medoids. If $NumRun = 1$, terminate the program; otherwise increment n and go to step 3.

3.3.2 Experimental Results

3.3.2.1 Test Datasets

Four artificial datasets and one real image dataset were used for the experiments as follows:

1. 1,500 objects collected from four elliptic clusters.

2. 12,000 objects collected from twelve elliptic clusters.

3. 3,100 objects collected from five compact clusters.

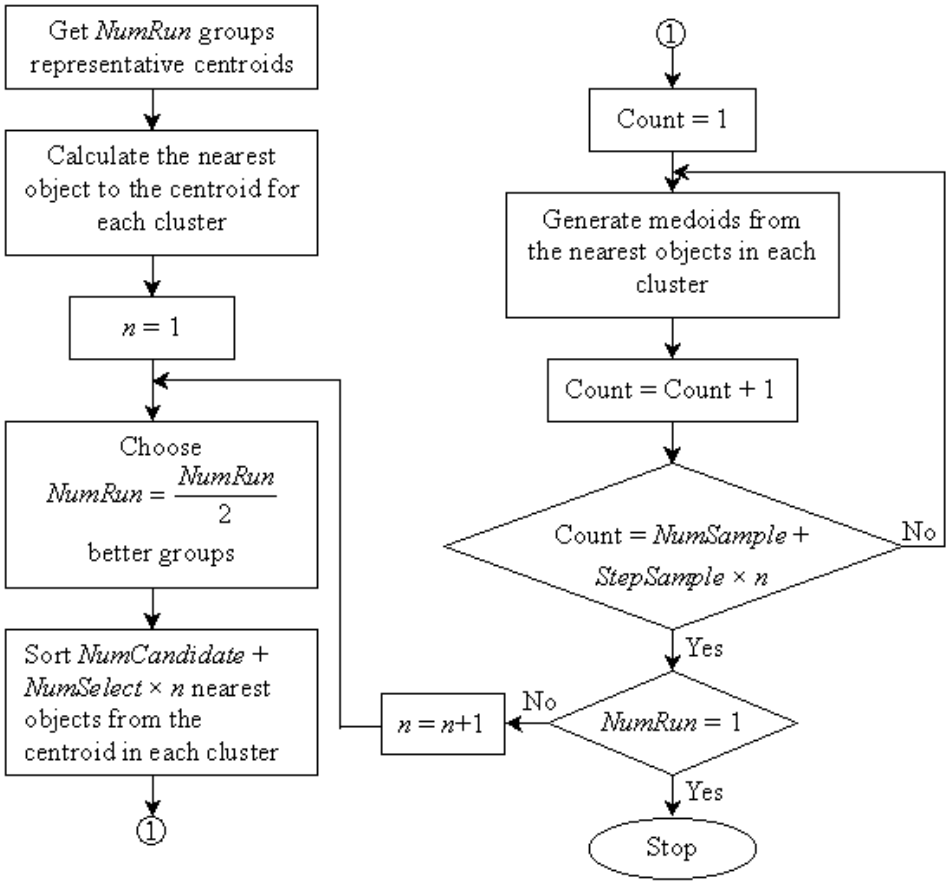


Figure 3.5: The flowchart of Incremental Multi-Centroid, Multi-Run Sampling Scheme ($IMCMRS$)

4. 3,000 objects with 8 dimensions were generated from the Gauss-Markov source that is of the form $y_n = \alpha y_{n-1} + w_n$ where w_n is a zero-mean, unit variance, Gaussian white noise process, with $\alpha = 0.5$.

5. 16,384 objects with 16 dimensions were generated from the $LENA$ grey-scale 512×512 image.

3.3.2.2 Implementation and Description

Experiments were carried out to test the number of distance calculations and the average distance per object for the $CLARA$, $CLARANS$, $MCMRS$ and $IMCMRS$ algorithms. Since the computation time depends not only on the

clustering algorithm but also on the use of computation facility. It is thus better to choose a measure criterion such that the measure results are the same for all types of computers and this measure criterion is proportional to the computation time – we chose the number of distance calculation as the benchmark. Squared Euclidean distance measure is used.

The four elliptic clusters were used for the first experiment and 12 medoids were selected from 1500 objects. For $CLARA$, the parameter q was set to 5 and s was set to $160+2k$. For $CLARANS$, the parameter *numlocal* was set to 5 and parameter *maxneighbor* was set to 270 (ie. 1.5% of $k \times t$). For $MCMRS$, k-means is used to generate 12 centroid-based clusters. The parameters $NumRun$, $NumCandidate$ and $NumSample$ in $MCMRS$ were set to 20, 10 and 200, respectively. For $IMCMRS$, k-means is also used to generate 12 centroid-based clusters. The parameters $NumRun$, $NumCandidate$, $NumSample$, $NumSelect$ and *StepSample* in $IMCMRS$ were set to 32, 1, 1, 1 and 5 respectively. The experimental results based on 10 runs for $CLARA$, $CLARANS$, $MCMRS$ and $IMCMRS$ are shown in Table 3.5 and Figure 3.6. Comparisons with $CLARA$ and $CLARANS$, show that $IMCMRS$ may reduce the computation time by more than 91% and 80%. Both $MCMRS$ and $IMCMRS$ performed better than $CLARANS$ and $CLARA$ both in the computation time and the average distance per object.

The twelve elliptic clusters were used for the second experiment. 12 medoids were selected from 12000 objects. For $CLARA$, the parameter q was set to 5 and s was set to $960+2k$. For $CLARANS$, the parameters *numlocal* and *maxneighbor* were set to 5 and 1800, respectively. For $MCMRS$, k-means is used to generate 12 centroid-based clusters. The parameters $NumRun$, $NumCandidate$ and $NumSample$ in $MCMRS$ were set to 20, 10 and 200, respectively. For $IMCMRS$, k-means algorithm is also used to generate 12 centroid-based clusters. The parameters $NumRun$, $NumCandidate$, $NumSample$, $NumSelect$ and *StepSample* in $IMCMRS$ were set to 32, 1, 1, 1 and 5 respectively. As shown

Table 3.5. Results of Experiment for Four Elliptic Clusters

seed	*CLARA*		*CLARANS*		*MCMRS*		*IMCMRS*	
	Average dist.	Count of dist.(10^5)	Average dist.	Count of dist.(10^5)	Average dist.	Count of dist.(10^5)	Average dist.	Count of dist.(10^5)
1	0.228	2004	0.228	1609	0.216	808	0.213	197
2	0.221	2302	0.232	1067	0.217	821	0.212	221
3	0.236	2388	0.236	1351	0.217	825	0.212	197
4	0.230	2686	0.233	1154	0.218	811	0.213	249
5	0.227	2430	0.232	1572	0.218	907	0.213	199
6	0.237	2260	0.227	1223	0.216	803	0.212	223
7	0.232	2646	0.234	895	0.217	821	0.213	244
8	0.236	2516	0.231	1429	0.217	803	0.213	232
9	0.233	2345	0.231	1148	0.216	803	0.213	194
10	0.238	2728	0.231	1167	0.218	805	0.212	215
Ave.	0.232	2431	0.232	1140	0.217	811	0.213	217

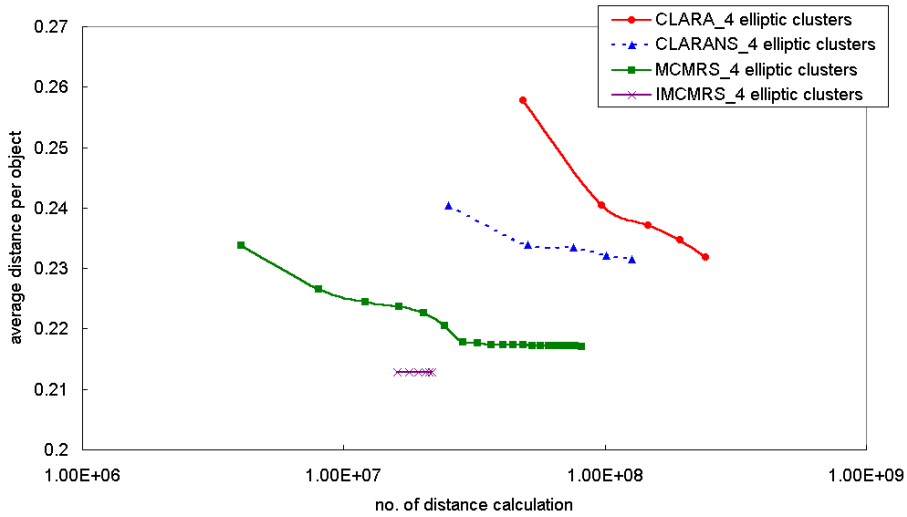


Figure 3.6: Performance Comparison of *CLARA*, *CLARANS*, *MCMRS* and *IMCMRS* for Four Elliptic Clusters

in Table 3.6 and Figure 3.7, *IMCMRS* will reduce the computation time by more than 98%, 97% and 73% by comparing with *CLARA*, *CLARANS* and *MCMRS*. Both *MCMRS* and *IMCMRS* are more efficient and effective than *CLARA* and *CLARANS*.

The compact clusters with noise were used for the third experiment. 5 medoids were selected from 3100 objects. For *CLARA*, the parameter q was set to 5

Table 3.6. Results of Experiment for Twelve Elliptic Clusters

seed	*CLARA*		*CLARANS*		*MCMRS*		*IMCMRS*	
	Average dist.	Count of dist.(10^5)	Average dist.	Count of dist.(10^5)	Average dist.	Count of dist.(10^5)	Average dist.	Count of dist.(10^5)
1	0.940	115659	0.931	91792	0.922	6587	0.927	1707
2	0.942	92528	0.943	63512	0.920	6554	0.920	1848
3	0.956	122462	0.935	76873	0.930	6496	0.920	1746
4	0.951	103413	0.928	79582	0.928	6447	0.939	1729
5	0.946	111577	0.949	90014	0.921	6483	0.918	1772
6	0.960	108856	0.936	72069	0.922	6632	0.919	1726
7	0.972	77562	0.936	84914	0.920	6426	0.918	1746
8	0.944	93889	0.934	93092	0.920	6549	0.920	1807
9	0.954	84365	0.930	59242	0.923	6573	0.922	1726
10	0.942	84365	0.931	75053	0.923	6541	0.920	1658
Ave.	0.951	99467	0.935	78615	0.923	6529	0.922	1747

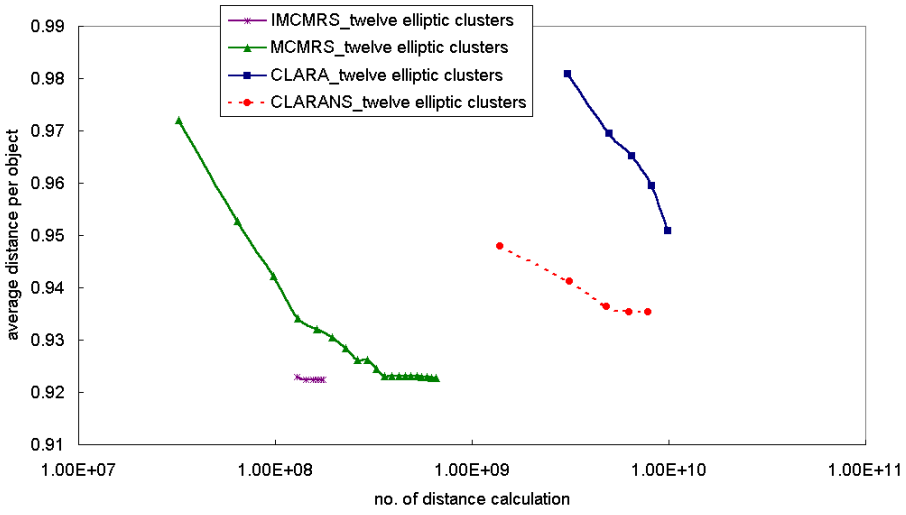


Figure 3.7: Performance Comparison of *CLARA*, *CLARANS*, *MCMRS* and *IMCMRS* for Twelve Elliptic Clusters

and s was set to $200 + 2k$. For *CLARANS*, the parameters *numlocal* and *maxneighbor* were set to 5 and 200, respectively. For *MCMRS*, k-means is used to generate 5 centroid-based clusters. The parameters *NumRun*, *NumCandidate* and *NumSample* in *MCMRS* were set to 20, 10 and 200, respectively. For *IMCMRS*, k-means is also used to generate 5 centroid-based clusters. The parameters *NumRun*, *NumCandidate*, *NumSample*, *NumSelect* and *StepSample*

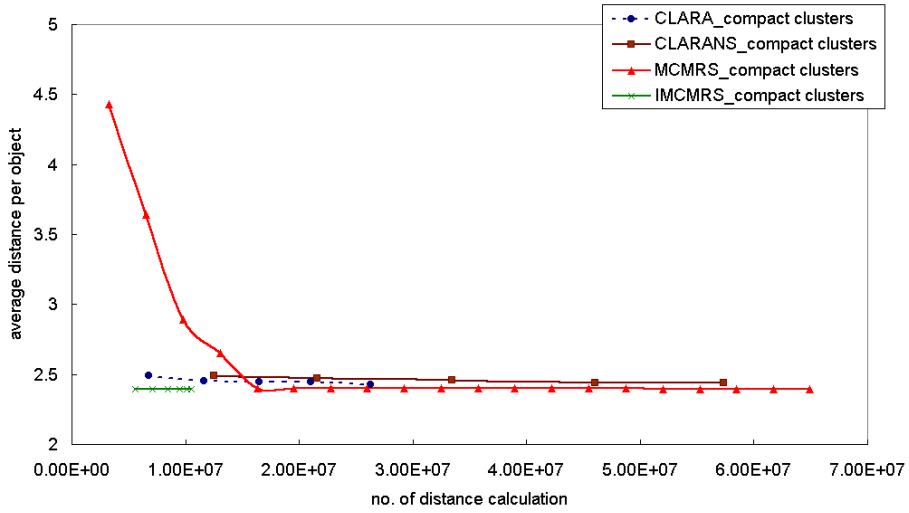


Figure 3.8: Performance comparison of $CLARA$, $CLARANS$, $MCMRS$ and $IMCMRS$ for Five Compact Clusters

in $IMCMRS$ were set to 32, 1, 1, 1 and 5 respectively. Experimental results based on 10 runs for $CLARA$, $CLARANS$, $MCMRS$ and $IMCMRS$ are shown in Table 3.7. If the database is not large and the medoid size is small, the performance of $CLARA$ is better than $CLARANS$ shown in Figure 3.8. Comparing with $CLARA$, $CLARANS$ and $MCMRS$ $IMCMRS$ may reduce the computation time by more than 60%, 81% and 83%, respectively.

Table 3.7. Results of Experiment for Compact Clusters

seed	$CLARA$		$CLARANS$		$MCMRS$		$IMCMRS$	
	Average dist.	Count of dist.(10^5)	Average dist.	Count of dist.(10^5)	Average dist.	Count of dist.(10^5)	Average dist.	Count of dist.(10^5)
1	2.436	253	2.432	584	2.398	646	2.397	99
2	2.425	274	2.457	600	2.397	647	2.397	107
3	2.430	264	2.429	604	2.398	655	2.397	96
4	2.440	295	2.442	504	2.397	649	2.397	102
5	2.405	232	2.431	583	2.398	651	2.397	103
6	2.411	253	2.419	606	2.398	655	2.397	102
7	2.435	243	2.457	530	2.397	642	2.397	104
8	2.422	285	2.470	519	2.397	654	2.397	104
9	2.444	253	2.417	620	2.397	647	2.397	112
10	2.440	274	2.424	581	2.398	645	2.397	120
Ave.	2.429	263	2.438	573	2.398	649	2.397	105

The Gauss-Markov source was used for the fourth experiment. 32 medoids were selected from 3000 objects. For $CLARA$, the parameter q was set to 5 and s was set to $320+2k$. For $CLARANS$, the parameters *numlocal* and *maxneighbor* were set to 5 and 1200, respectively. For $MCMRS$, k-means algorithm is used to generate 32 centroid-based clusters. The parameters $NumRun$, $NumCandidate$ and $NumSample$ in $MCMRS$ were set to 20, 10 and 200, respectively. For $IMCMRS$, k-means is also used to generate 32 centroid-based clusters. The parameters $NumRun$, $NumCandidate$, $NumSample$, $NumSelect$ and $StepSample$ in $IMCMRS$ were set to 32, 1, 1, 1 and 5 respectively. Experimental results are shown in Table 3.8 and Figure 3.9, comparing with $CLARA$ and $MCMRS$, $IMCMRS$ may reduce the computational complexity by more than 99% and 70%, respectively. The proposed $IMCMRS$ can reduce the computation time by approximately a factor of 30 and also obtains a better average distance in comparison with $CLARANS$.

Table 3.8. Results of Experiment for Gauss-Markov Source

seed	*CLARA*		*CLARANS*		*MCMRS*		*IMCMRS*	
	Average dist.	Count of dist.(10^5)	Average dist.	Count of dist.(10^5)	Average dist.	Count of dist.(10^5)	Average dist.	Count of dist.(10^5)
1	4.559	154809	4.432	37604	4.487	4439	4.357	1340
2	4.592	163692	4.359	53234	4.476	4349	4.382	1330
3	4.551	178918	4.381	37512	4.490	4411	4.376	1273
4	4.578	172574	4.398	40559	4.495	4417	4.371	1364
5	4.559	182725	4.367	39694	4.489	4403	4.375	1234
6	4.526	167498	4.384	41370	4.489	4357	4.379	1278
7	4.527	185263	4.380	32312	4.485	4379	4.377	1309
8	4.483	162423	4.394	39600	4.499	4382	4.361	1307
9	4.545	190338	4.377	36707	4.491	4400	4.369	1276
10	4.514	180187	4.406	35835	4.485	4378	4.329	1309
Ave.	4.543	173843	4.388	39443	4.489	4391	4.368	1302

The *Lena* grey-level image data with size 512 by 512 is used for the fifth experiment. 16384 objects with 16 dimensions were extracted from this image. 8 medoids were selected from these 16384 objects. For $CLARA$, the parameter q was set to 5 and s was set to $1000+2k$. For $CLARANS$, the parameters *numlocal*

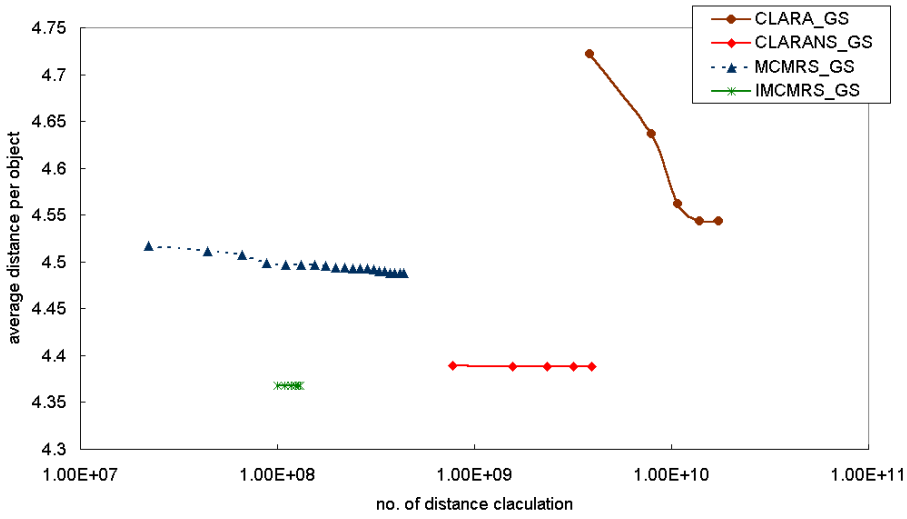


Figure 3.9: Performance Comparison of $CLARA$, $CLARANS$, $MCMRS$ and $IMCMRS$ for Gauss-Markov Source

and *maxneighbor* were set to 5 and 1800, respectively. For $MCMRS$, k-means algorithm is used to generate 8 centroid-based clusters. The parameters $NumRun$, $NumCandidate$ and $NumSample$ in $MCMRS$ were set to 20, 10 and 200, respectively. For $IMCMRS$, k-means is also used to generate 8 centroid-based clusters. The parameters $NumRun$, $NumCandidate$, $NumSample$, $NumSelect$ and $StepSample$ in $IMCMRS$ were set to 32, 1, 1, 1 and 5 respectively. Experimental results based on 10 runs for $CLARA$, $CLARANS$, $MCMRS$ and $MCMRS$-$CLASA$ are shown in Table 3.9 and Figure 3.10, comparing with $CLARA$ and $MCMRS$, $IMCMRS$ may reduce the computational complexity by more than 93% and 69%, respectively. The proposed $IMCMRS$ can reduce the computation time by approximately a factor of 33 and also get better average distance compared with $CLARANS$.

Table 3.9. Results of Experiment for *Lena* Image

seed	*CLARA*		*CLARANS*		*MCMRS*		*IMCMRS*	
	Average dist.	Count of dist.(10^5)	Average dist.	Count of dist.(10^5)	Average dist.	Count of dist.(10^5)	Average dist.	Count of dist.(10^5)
1	2829.56	30057	2816.43	79704	2800.15	6121	2796.00	1869
2	2833.36	31379	2831.77	56796	2802.49	5897	2795.52	1781
3	2839.22	32700	2806.48	76182	2801.81	6138	2796.36	1760
4	2821.52	36003	2814.16	59684	2801.98	6080	2796.62	1822
5	2827.43	28736	2821.41	44216	2802.71	6067	2796.00	1925
6	2842.22	28736	2822.10	54274	2801.52	6158	2796.44	1801
7	2838.23	30718	2817.13	55947	2800.16	6145	2796.54	1892
8	2843.91	28736	2819.38	56845	2803.82	6036	2795.62	1862
9	2820.76	24772	2823.58	64621	2802.98	6229	2796.36	2097
10	2819.52	28736	2826.95	59039	2901.34	6019	2796.80	1592
Ave.	2831.57	30057	2819.94	60731	2801.90	6089	2796.14	1840

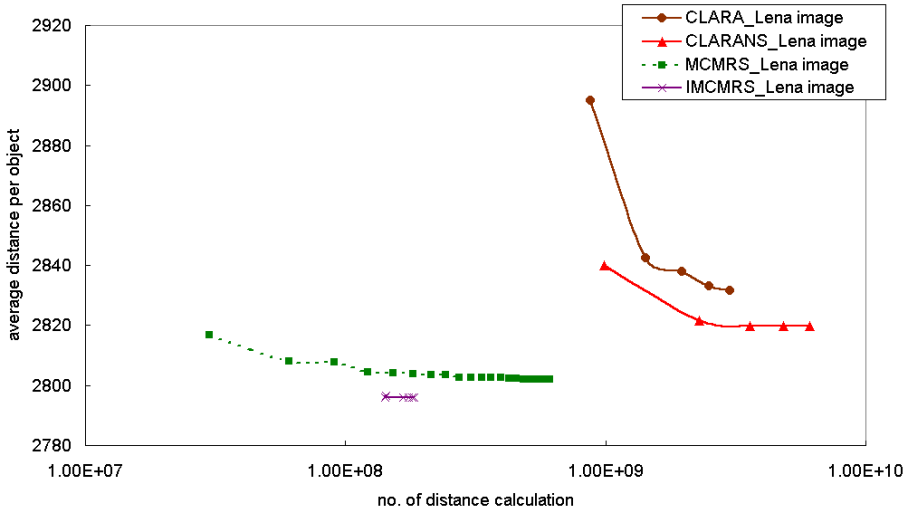


Figure 3.10: Performance Comparison of *CLARA*, *CLARANS*, *MCMRS* and *IMCMRS* for *Lena* Image

Chapter 4

Improved Centroid-Based Clustering Algorithms

The goal of clustering is to group sets of objects into classes such that homogeneous objects are placed in the same cluster while dissimilar objects are in separate clusters. There are three important evaluation criteria for centroid-based cluster generation as follows:

- To reduce the average distortion for each pattern in the training database.
- To get the high classification rate for test database.
- To reduce the computational complexity.

One of the most well known techniques for centroid-based clustering is the k-means algorithm (or GLA, or LBG) (MacQueen 1967, Linde et al. 1980). Unfortunately, it is a descent algorithm and often obtains the local optimum. In order to improve the performance, the simulated annealing method (Vaisey & Gersho 1988), the tabu search approach (Al-Sultan 1995) and tabu search with GLA method (Franti et al. 1998) were proposed. In this chapter, the tabu search approach combining with the simulated annealing algorithm for cluster generation will be proposed in section 4.2. The main idea is to modify the tabu search

approach by introducing the counter to limit the non-local moves from the current best solution and forbid the current best solution to keep the same more than some fixed iterations. Especially, the simulated annealing method is applied to select the suitable current best solution so that the performance is improved compared with the tabu search approach with GLA algorithm for cluster generation (Chu & Roddick 2000), (Chu & Roddick 2003).

K-means and some related algorithms are effective for uniformly distributed data, but it is not effective for non-uniformly distributed data. In section 4.3, the genetic algorithm with stochastic relaxation approach on mutation operator was developed for clustering and was also applied to the codebook design of mean-residual vector quantization. An incremental splitting clustering algorithm is presented in section 4.4 to improve the centroid-based clustering for non-uniformly distributed data. In addition, we explained a novel labeled bisecting k-means clustering algorithm and applied to the robust image watermarking depicted in section 4.5. Finally, an efficient nearest neighbour clustering algorithm based on Hadamard transform was proposed and applied to the codeword search for vector quantization, which was presented in section 4.6. In the following section, we will review some centroid-based clustering algorithms.

4.1 Related Existing Centroid-Based Clustering Algorithm

The existing clustering algorithms can be simply classified into the following two categories: hierarchical clustering and partitional clustering (Jain & Dubes 1988). The hierarchical clustering operates by partitioning the patterns into successively fewer structures. Hierarchical procedures can be either agglomerative or divisive. Partitional clustering procedures typically start with the patterns partitioning into a number of clusters and divide the patterns by increasing the

number of partitions. The most popular class of partitional clustering methods are the prototype-based clustering algorithms. In the prototype-based clustering algorithms, each cluster in represented by a prototype, and the sum of distance from the pattern to the prototype is usually used as the objective function. If the prototype of one cluster is the centre of this cluster, it is called centroid-based cluster. Normally, the prototypes are the centres of the clusters.

4.1.1 K-Means (GLA)

K-means (or GLA, or LBG) (Linde et al. 1980) is a centroid-based cluster algorithm. It is a descent algorithm in the sense that the average distortion is reduced at each iteration. For this reason, the k-means algorithm tends to get trapped in local optima. The performance of the k-means algorithm depends on the number of optima and the choice of the initial condition. The k-means (or GLA, or LBG) algorithm can be described as follows:

Step 1. Select N clustering data vectors as the initial centroids of the clusters randomly, where N is the number of clusters (or codebook size). Set $n = 0$, where n is the iteration count.

Step 2. Find the nearest centroid to each clustering data vector. Put X_j in the partitioned set (or cluster) P_i if C_i is the nearest centroid to X_j.

Step 3. After obtaining the partitioned sets $P = (P_i; 1 \leq i \leq N)$, increment n. Calculate the overall average distortion

$$D_n = \frac{1}{T} \sum_{i=1}^{N} \sum_{j=1}^{T_i} D(X_j^{(i)}, C_i)$$

where $P_i = \{X_1^{(i)}, X_2^{(i)}, \ldots, X_{T_i}^{(i)}\}$, T_i is the number of clustering data vectors belonging to the partitioned set P_i.

Step 4. Find centroids of all disjoint partitioned sets P_i by

$$C_i = \frac{1}{T} \sum_{j=1}^{T_i} X_j^{(i)}$$

Step 5. If $\frac{D_{n-1}-D_n}{D_n} > \epsilon$, go to step 2; otherwise terminate the program. ϵ is a small distortion threshold.

4.1.2 Simulated Annealing for Clustering

Simulated annealing is a metaheuristic recently proposed by Kirkpatrick *et. al* (Kirkpatrick et al. 1983) specifically for combinatorial optimization problems. Simulated annealing is termed a metaheuristic because it can be combined with other heuristic procedures to prevent them from trapping at local optima. Vecchi and Kirkpatrick applied a simulated annealing method to the optimization of a wiring problem (Vecchi & Kirkpatrick 1983). Gamal et al. also used the method of simulated annealing to construct good source codes, error-correcting codes and spherical codes (Gamal et al. 1987). The simulated annealing was also applied to cluster generation for vector quantization (Cetin & Weerackody 1988, Vaisey & Gersho 1988). The simulated annealing for clustering patterns can be described as follows:

Step 1. The training pattern $X_j, j = 1, 2, \ldots T$, is partitioned into the cluster $S_i, i = 1, 2, \ldots N$ randomly. Set $n = 0$ and calculate the center of the cluster.

$C_i = \frac{1}{|S_i|} \sum_{X_j \in S_i} X_j$

where $|S_i|$ denotes the number of training patterns in the cluster S_i.

Step 2. The clusters are perturbed by randomly selecting a pattern and moving this pattern from its current cluster to the different randomly selected cluster. Calculate the new centroids.

Step 3. The change in distortion ΔD is defined as the distortion of current clusters minus the distortion of previous clusters. The perturbation is accepted if $e^{-\frac{\Delta D}{\hat{T}_n}} > \gamma$, where γ is a random value generated uniformly on the interval $[0, 1]$. $\hat{T}_n$ is nth iterative temperature.

Step 4. If the distortion of the current clusters reaches the desired value or the iterative number n reaches the predetermined value, then terminate the program; otherwise, set $n = n + 1$ and go to step 2.

The simulated annealing algorithm starts with an initial temperature $\hat{T}_0$. The temperature sequence $\hat{T}_0, \hat{T}_1, \hat{T}_2 \cdots$ are positive numbers which is called an annealing schedule where

$\hat{T}_0 \geq \hat{T}_1 \geq \hat{T}_2 \cdots$

and

$\lim_{n\to\infty} \hat{T}_n = 0$.

4.1.3 Tabu Search Approach for Clustering

Al-Sultan (Al-Sultan 1995) applied the tabu search approach to cluster the patterns. A set of test solutions is generated from the current solution randomly. For each pattern, a random number, $0 \leq R \leq 1$, is generated. If $R \geq P_t$, then this pattern is assigned to cluster i, where i is randomly generated but not the same cluster as in the current solution, $1 \leq i \leq N$, and P_t is the predefined probability threshold; otherwise it is partitioned to the same cluster as in the current best solution. Starting with the best solution and progressing to the worst one, if the aspiration level is satisfied or the tabu conditions are avoided, then this test solution is chosen as the current best solution and each pattern assigned to the ith cluster in the current best solution is recorded in the tabu list memory. If all solutions are tabu, then new solutions are generated from the current best solution again. The program is terminated if the predefined distance or the maximum number of iterations is reached. The tabu search approach for clustering patterns in (Al-Sultan 1995) is illustrated in Figure 4.1.

The tabu search approach is also combined with the k-means (or GLA) algorithm for cluster generation (Franti et al. 1998) and the centers of the clusters are taken as the codevectors for vector quantization. The performance comparison of

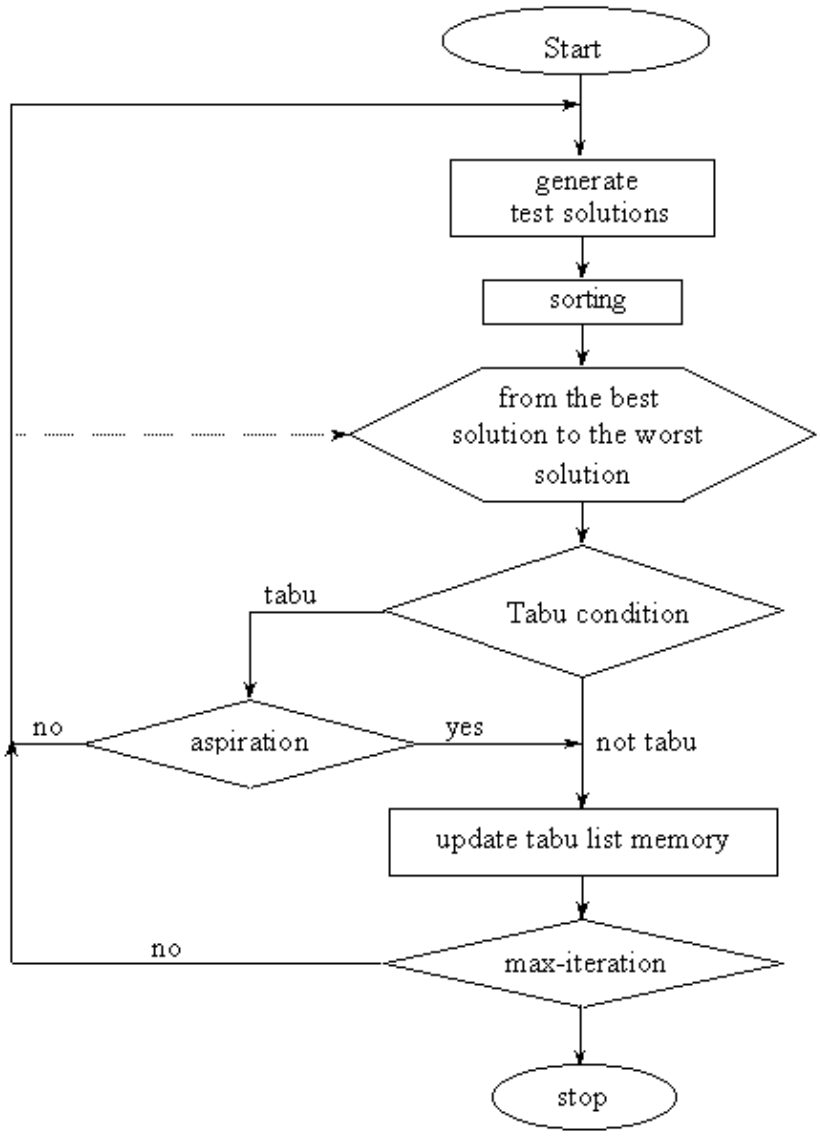


Figure 4.1: Flowchart of The Tabu Search Approach for Clustering Patterns

this algorithm is better than both GLA algorithm and the tabu search approach for cluster generation. In (Franti et al. 1998), there are two types of test solutions: partition based test solution and codebook based test solution. In partition based test solution, all the training patterns form the test solutions. In codebook based test solution, the elements of the test solution are the centers of the clusters. The basic procedure of the tabu search approach with GLA algorithm for clustering patterns is illustrated in Figure 4.2.

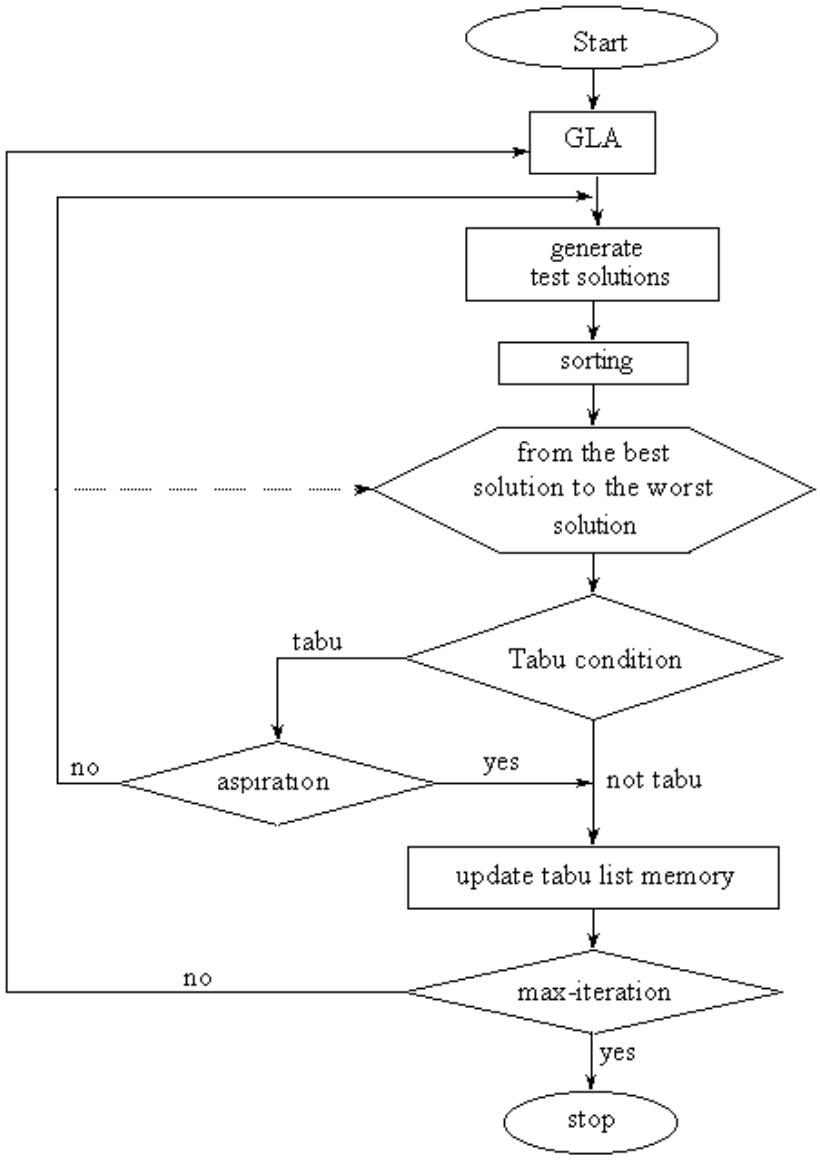


Figure 4.2: Flowchart of The Tabu Search Approach with GLA Algorithm

4.1.4 Clustering Using Stochastic Relaxation Approach

The simulated annealing for clustering is to perturb the centers of the clusters by moving the training pattern from its current cluster to a different cluster. The perturbation is accepted based on the probability of the generated random number, the change in distortion and current temperature. For stochastic relaxation approach (Zeger & Gersho 1989, Zeger, Vaisey & Gersho 1992), the parturbation is applied by adding some values to the centers of the clusters definitely for each iteration. The stochastic relaxation approach for clustering can be described as follows:

Step 1. Randomly select initial centers of clusters $C_i^{(1)}, i = 1, 2, \ldots N$, N is the number of clusters. Set iterative number $n = 1$ and $D_0 = \infty$.

Step 2. Assign the training data X_p to the ith cluster if $d(X_p, C_i) \leq d(X_p, C_j)$, $i \neq j$, $j = 1, 2, \ldots N$. Calculate the overall distortion $D_n = D_n + \|X_p - C_i\|^2$.

Step 3. If $\frac{|D_{n-1} - D_n|}{D_n} < \epsilon$, then terminate the program; otherwise, set $n = n + 1$.

Step 4. Calculate the centers of clusters $C_i = \frac{1}{|S_i|} \sum_{X_p \in S_i} X_p$, $i = 1, 2, \ldots N$. where $|S_i|$ denotes the member of training patterns in the ith cluster.

Step 5. Perturb the centers of clusters using $C_i^{(n)} = C_i^{(n)} + \delta_i(T_n)$. Go to Step 2.

$\delta_i(T_n)$ is a prturbation function where the value of the temperature T_n decreases with the increase of the iterative number n. $\delta_i(T_n)$ can be a uniform distribution with zero mean and T_n is the range (Zeger & Gersho 1989, Zeger et al. 1992). T_n can be $\sigma_x^2 \alpha^n$, $\alpha = 0.95$, σ_x^2 is the variance of the training patterns.

4.2 Clustering using Tabu Search with Simulated Annealing

4.2.1 Motivation

A cluster generation for vector quantization using the tabu search approach with simulated annealing was proposed (Chu & Roddick 2003). The main idea of this algorithm is to use the tabu search approach to generate non-local moves for the clusters and apply the simulated annealing technique to select current best solution, thus improving the time taken for cluster generation and reducing the mean squared error. Although the tabu search approach can avoid the cycling condition so that jumps out of local optimum, it can be further improved by introducing counters to calculate the frequency of the move for each element, i.e., the non-local moves from the current best solution can be limited by counting the frequency of the move. The idea is to keep the movement of each element within the same probability to reduce the probability of cycling. This is done as sometimes the est solution does not change for long periods which reduces the clustering performance. Simulated annealing is used to decide which test solution is suitable to be the current best solution for generating the test solutions for next iteration. The proposed tabu search approach with simulated annealing algorithm for cluster generation is shown in Figure 4.3 and described as follows:

Step 1. Generate an initial solution C_{init} using GLA algorithm. Set C_{curr}=C_{best}=C_{init}. Set a counter $Count_j$ for each element in the solution, $j = 1, 2, \ldots T$. T is the total number of training patterns.

Step 2. Generate S test solutions C_i by changing the values of elements from the current best solution C_{curr} with probability P_t. For each test solution, if the qth element of the test solution is changed and the value of $Count_q$ is smaller than ν, then increment $Count_q$. If all the values of the counters

are equal to ν, then reset all counters to zero.

Step 3. Calculate the mean squared error $D(C_i)$ for each test solution and sort these test solutions using mean squared error in increasing order.

Step 4. From the best test solution to the worst test solution, if $D(C_{best}) > D(C_i)$, set $C_{curr} = C_i$ and go to step 6; otherwise go to next step.

Step 5. Calculate $\Delta D = D(C_{curr}) - D(C_i)$ and set $\hat{T}_n = \hat{T}_0\alpha^n$. Generate a random number γ $(0 \leq \gamma \leq 1)$, if $\gamma < e^{\frac{\Delta D}{\hat{T}_n}}$, then set $C_{curr} = C_i$ and go to step 6, else if $i = S$, then go to step 2; otherwise increment i and go to step 4.

Step 6. If $D(C_{best})$ remains unchanged for M iterations, then set C_{curr}=C_{best}, clear the counter $Count_j$, $j = 1, 2, \ldots T$. If $D(C_{best}) > D(C_{curr})$ then set $C_{best} = C_{curr}$. If the number of iteration has reached, then terminate the program; otherwise go to step 2.

4.2.2 Experimental Results

Experiments were carried out to test the clustering distortion using the GLA algorithm (Linde et al. 1980), the tabu search approach with GLA algorithm (Franti et al. 1998) (referred to as $TABU-GLA$), and our proposed tabu search approach with simulated annealing (referred to as $TABU - SA$) (Chu & Roddick 2003).

The initialization of the tabu search approach with simulated annealing can be randomly generated or obtained from the GLA algorithm. Since the codebook based test solution for the tabu search approach with GLA algorithm is better than the partitioned based test solution in (Franti et al. 1998), we adopt the codebook based test solution for the tabu search approach with GLA algorithm for cluster generation. The evaluation function of the clustering distortion is the

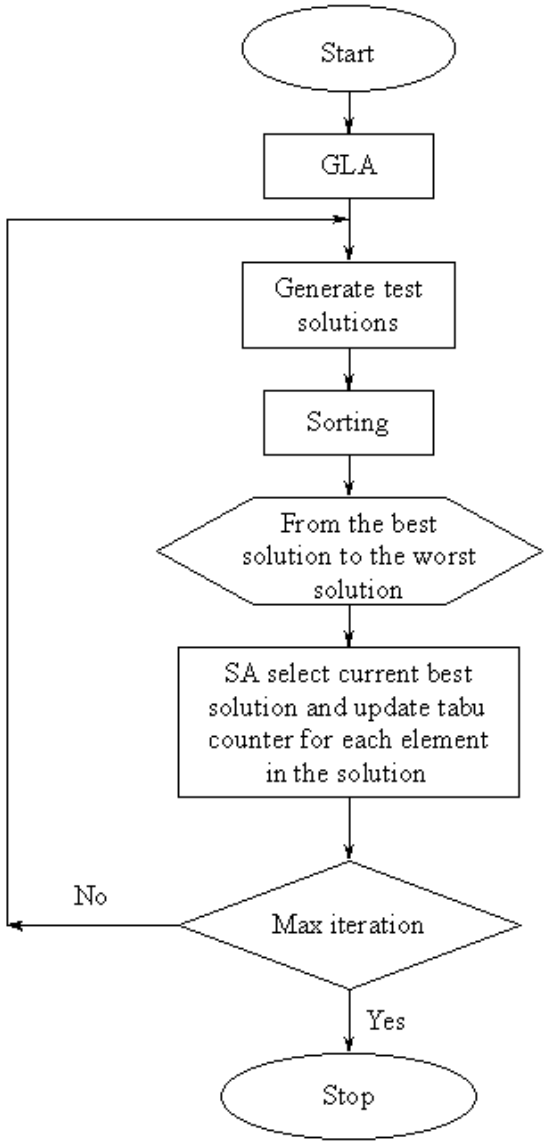


Figure 4.3: Flowchart of The Tabu Search Approach with Simulated Annealing

mean squared error (MSE) as following:

$$MSE = \frac{1}{kT} \sum_{i=1}^{N} \sum_{j=1}^{T_i} D(X_j^{(i)}, C_i)$$

where T_i is the number of clustering patterns belonging to the ith cluster. T is the total number of training patterns and k is the number of dimension for each pattern. $X_j^{(i)}$ is the jth pattern belonging to the ith cluster.

Three 512×512 images, the "*Lena*", "*Baboo*", and "*Pepper*" images were

used as the test material. 4×4 pixel blocks were taken from these gray images as the training patterns, i.e., the number of dimensions was 16. The parameters used for the test solutions S, number of training patterns T, probability threshold P_i, the limit value of counter ν, the limit number of the same current best iterations, the initial temperature $\hat{T}_0$ and the value of α were 20, 16384, $\frac{100}{16384}$, 3, 30, 500 and 0.99, respectively.

Experimental results shown from Table 4.1 to Table 4.3 and Figure 4.4 to Figure 4.6 demonstrate that, at the 1 hour mark, the proposed algorithm can reduce the average distortion by $0.3\% \sim 9.6\%$ and $10\% \sim 13\%$ compared with the GLA algorithm and tabu search approach with GLA algorithm, respectively. After 8 hours, the proposed algorithm will reduce the average distortion by $1.67\% \sim 9.99\%$ and $1.65\% \sim 5.15\%$ comparing with the GLA algorithm and tabu search approach with the GLA algorithm, respectively. The original image and decoded images using GLA, $TABU - GLA$, $TABU - SA$ with random initialization, and $TABU - SA$ with GLA initialization are shown in Figure 4.7, 4.8 and 4.9.

Table 4.1: Performance Comparison Using *Lena* Image as The Training Patterns

Algorithm	TS-SA with GLA initialization	TS-SA with random initialization	GLA	TS-GLA
Initialization	92.476	99.403	72.360	68.634
1 hour	55.912	57.354	59.889	64.309
2 hour	55.431	56.310	59.889	60.234
3 hour	55.431	56.129	59.889	59.445
4 hour	55.431	56.129	59.889	59.280
5 hour	55.431	56.129	59.889	59.216
6 hour	55.431	56.129	59.889	59.018
7 hour	55.431	56.129	59.889	58.501
8 hour	55.431	56.129	59.889	58.434

Preliminary experimental results demonstrate the proposed tabu search ap-

Table 4.2: Performance Comparison Using *Baboo* Image as The Training Patterns

Algorithm	TS-SA with GLA initialization	TS-SA with random initialization	GLA	TS-GLA
Initialization	355.181	360.119	330.381	323.543
1 hour	308.624	305.591	309.636	312.276
2 hour	306.797	304.385	309.636	311.043
3 hour	305.628	304.141	309.636	309.958
4 hour	305.005	303.492	309.636	309.693
5 hour	304.575	303.218	309.636	309.298
6 hour	304.399	302.906	309.636	309.264
7 hour	304.304	302.772	309.636	309.264
8 hour	304.158	302.695	309.636	309.264

Table 4.3: Performance Comparison Using *Pepper* Image as The Training Pattern

Algorithm	TS-SA with GLA initialization	TS-SA with random initialization	GLA	TS-GLA
Initialization	127.286	140.189	82.332	75.351
1 hour	60.328	61.788	66.788	67.041
2 hour	60.117	60.891	66.788	66.005
3 hour	60.117	60.891	66.788	64.969
4 hour	60.117	60.891	66.788	64.409
5 hour	60.117	60.891	66.788	64.185
6 hour	60.117	60.891	66.788	63.804
7 hour	60.117	60.891	66.788	63.496
8 hour	60.117	60.891	66.788	63.381

proach with simulated annealing algorithm for cluster generation is superior to the tabu search approach with Generalised Lloyd algorithm (GLA).

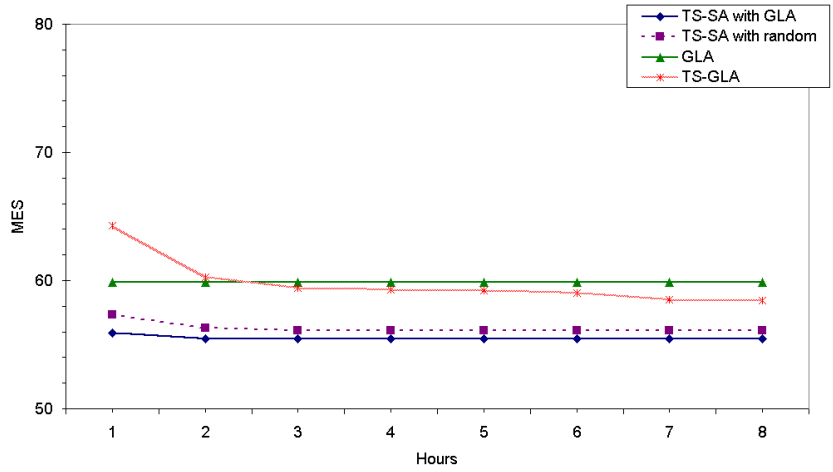


Figure 4.4. Performance Comparison Using *LENA* Image

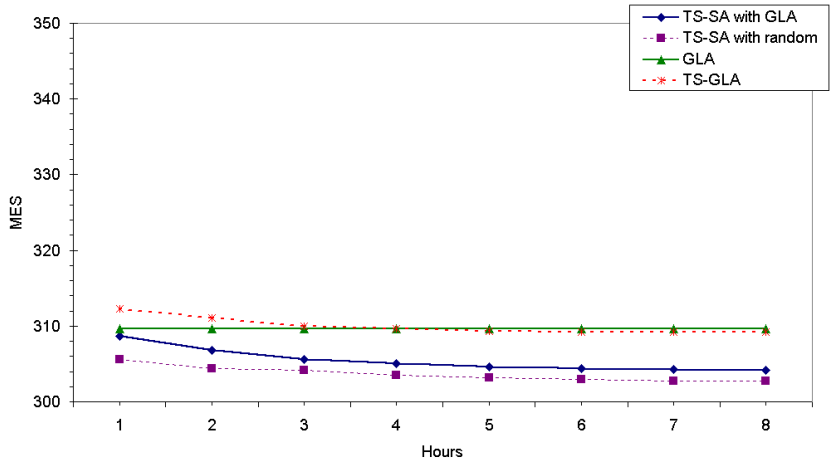


Figure 4.5. Performance Comparison Using *Baboo* Image

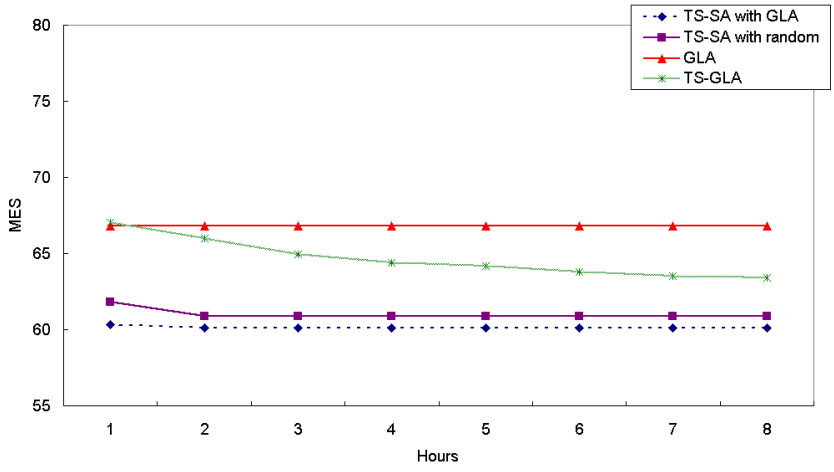


Figure 4.6. Performance Comparison Using *Pepper* Image

Figure 4.7: Comparison of *Lena* Image Recovered by Different Algorithm

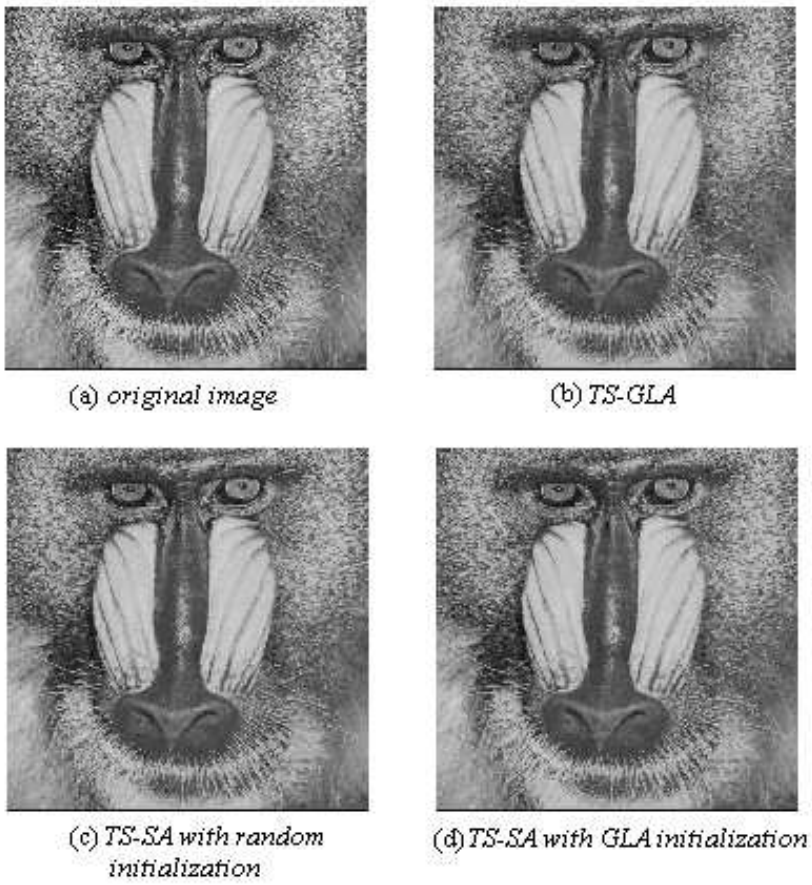


Figure 4.8: Comparison of *Baboo* Image Recovered by Different Algorithm

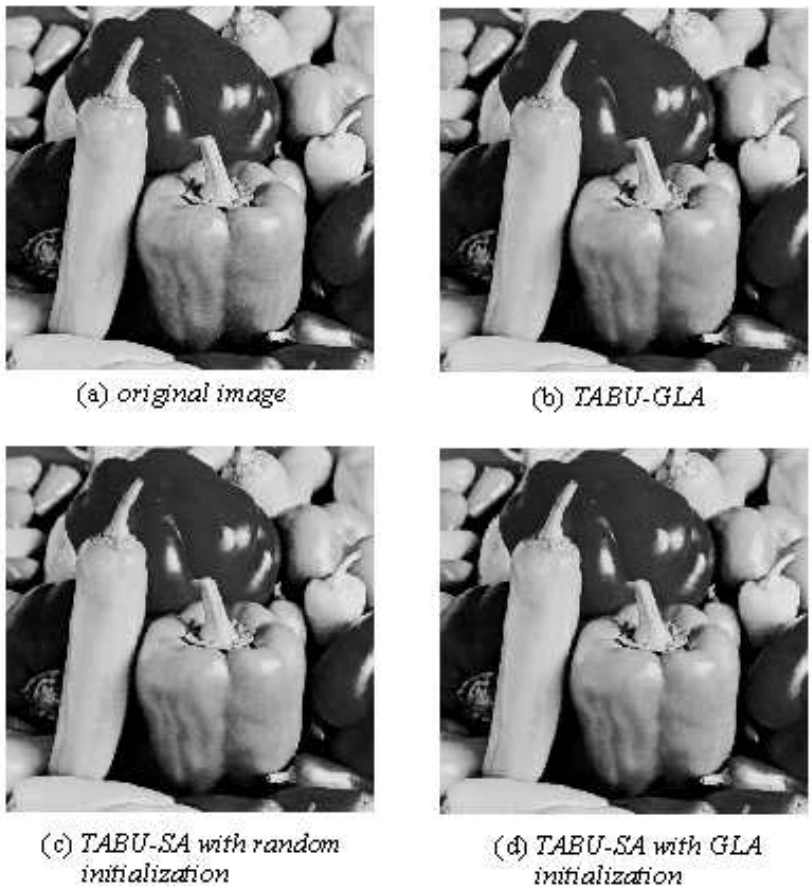


Figure 4.9: Comparison of *Pepper* Image Recovered by Different Algorithm

4.3 Genetic Clustering for Mean-Residual Vector Quantization

4.3.1 Motivation

Vector quantization (VQ) has been shown to be effective for pattern recognition, image coding and data compression (Gersho & Gray 1992). It is shown in Figure 4.10, the encoder of VQ encodes a given set of k-dimensional data patterns

$$X = \{X_j \mid X_j \in R^k; j = 1, 2, \ldots T\}$$

with a much smaller set of codevectors

$$C = \{C_i \mid C_i \in R^k; i = 1, 2, \ldots N\}$$

, where T is total number of data patterns and N is the codebook size. The compression operator by storing or sending the index only. The decoder has the same codebook as the encoder and decoding is operated by table look-up procedure. The performance of the data compression depends on good representative codevectors. Normally the representative codevectors are the centroids of the clusters generated from the training data patterns.

The mean-residual vector quantization (M/R VQ) (Gersho & Gray 1992) is a two-stage vector quantization that takes the mean and residual values to effect the vector quantization. It consists of the mean codebook (or mean scalars) and the residual codebook (or residual vectors). Assuming the data vector $X = \{x_1, x_2, \ldots, x_k\}$ and the code vector $C_i = \{c_{i1}, c_{i2}, \ldots, c_{ik}\}$, the squared Euclidean distortion measure can be expressed as

$$D(X, C_i) = \sum_{j=1}^{k} (x_j - c_{ij})^2. \tag{4.1}$$

The index i (compressed data) can be obtained from the nearest neighbor search and

$$i = argmin_p D(X, C_p). \tag{4.2}$$

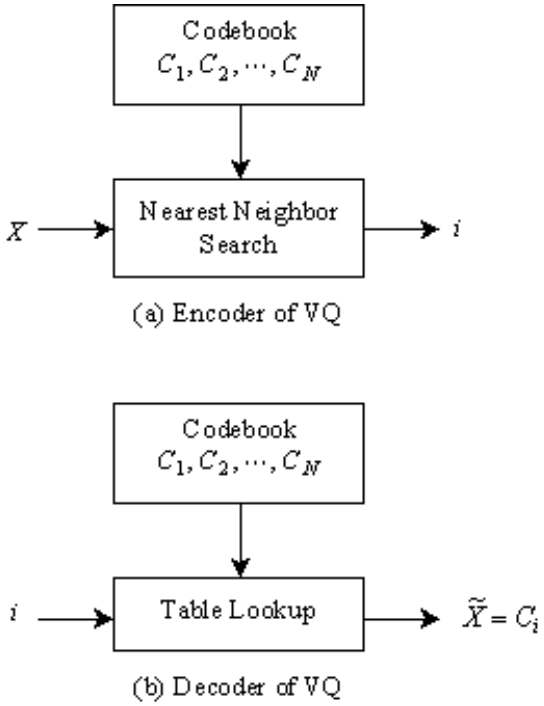


Figure 4.10. Block Diagram of *VQ* Compression

Assuming $\tilde{X}_j$ is the recovered data vector for the data vector X_j, $j = 1, 2, \ldots, T$, T is the total number of input vectors. The benchmark for evaluating the performance of data compression can be mean squared error (MSE), signal to noise ratio (SNR) and the peak signal to noise ratio ($PSNR$) as follows:

$$MSE = \frac{\sum_{j=1}^{T} D(X_j, \tilde{X}_j)}{T} \tag{4.3}$$

$$SNR = \frac{\sum_{j=1}^{T} X_j^2}{\sum_{j=1}^{T} D(X_j, \tilde{X}_j)} \tag{4.4}$$

$$PSNR = 10\ log \frac{(peak\ signal)^2}{MSE} \tag{4.5}$$

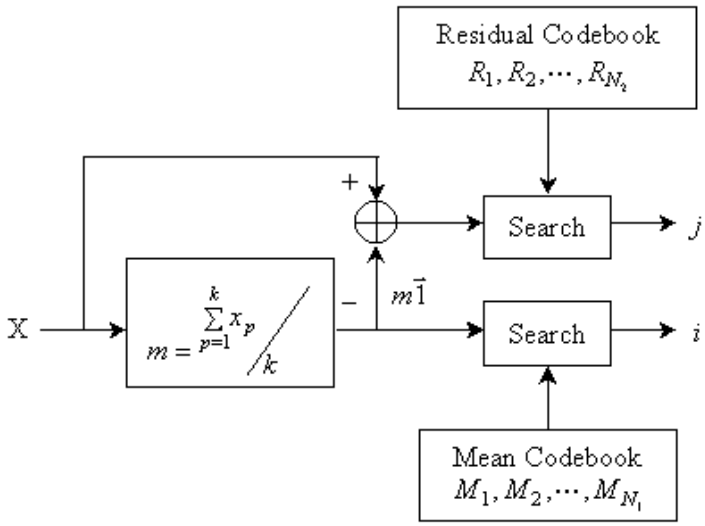
Obviously, the performance of the data compression depends on good code vectors.

The main benefit of *M/R VQ* is to reduce the storage of the codebook and the encoding time with only a small degradation in the recovery performance. As shown in Figure 4.11, the mean is calculated from the input vector $X = \{x_1, x_2, \ldots, x_k\}$ and the residual vector $X - m \cdot \vec{1} = \{x_1 - m, x_2 - m, \ldots, x_k - m\}$

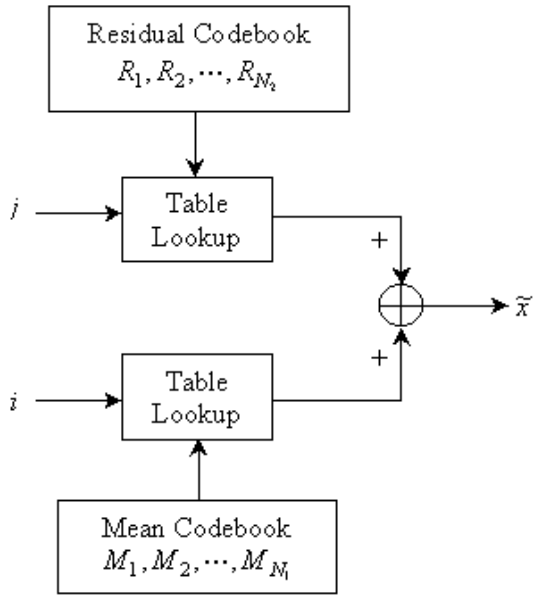
is obtained by subtracting the mean m from the input vector X. The mean index i and the residual index j are obtained by using the nearest neighbor search from the mean codebook and residual codebook, respectively. The indices i and j are the compressed data vector which may be stored as the database or sent to the receiver, and the recovered vector $\tilde{X} = \{M_i + r_{j_1}, M_i + r_{j_2}, \ldots, M_i + r_{j_k}\}$ can be obtained by a table lookup procedure from the mean codebook and residual codebook.

The GLA algorithm (Linde et al. 1980) is the most popular algorithm for designing the codebook from training data. Genetic algorithms (Delport & Koschorreck 1995, Pan, McInnes & Jack 1995) and simulated annealing methods (Vaisey & Gersho 1988, Flanagan, Morrell, Frost, Read & Nelson 1989, Lu & Morrell 1991) have been applied to generate better codebooks of single stage VQ with more computation time. In (Pan et al. 1996*b*), several fast clustering algorithms are presented by using the previous vector candidate, partial distance search, triangular inequality elimination and improved absolute error inequality criterion to design the codebook. A more promising algorithm combined with the genetic algorithm and simulated annealing method is proposed not only to reduce the peak signal to noise ratio but also to accelerate the design of the codebook (Huang, Pan, Lu, Sun & Hang 2001). There much research on codebook design of the single stage vector quantization, but, as yet, no advanced report on the codebook design of mean-residual vector quantization.

Genetic algorithms (Goldberg 1989) are an effective, parallel and near global optimum search method based on the ideas found in natural selection and genetics. During the search process, it can automatically achieve and accumulate the knowledge about the search space, and adaptively control the search process to approach a globally optimal solution. As shown in Figure 4.12, the GLA algorithm is applied to cluster the mean scalars and residual vectors (for $MaxIte$ iterations) separately. In (Chu & Roddick 2002, Chu, Roddick & Chen 2004), genetic algorithm with stochastic relaxation approach is used in combination with



(a) Encoder of M/R VQ



(b) Decoder of M/R VQ

Figure 4.11: Schematic for Mean-Residual Vector Quantization (*M/R VQ*)

the *GLA* algorithm to cluster the data vectors for mean-residual vector quantization. The chromosome consists of the mean code scalars and residual code vectors. The operators of selection, crossover and mutation are used on both the mean code scalars and residual code vectors simultaneously. The fitness function is the inverse of the mean squared error (or peak signal to noise ratio) based on

the original data vectors and recovered data vectors. After running the genetic algorithm, the GLA algorithm is applied again. The procedure continues until the maximum number of iterations ($MaxGen$) is reached or some quality metric is satisfied. The proposed algorithm can be described as follows:

Step 0. Calculate the mean scalars

$$m_i = \frac{\sum_{p=1}^{k} x_{ip}}{k}$$

and residual vectors $r_i = \{r_{i1}, r_{i2}, \ldots r_{ik}\}$, $r_{ij} = x_{ij} = m_i$ from the training data vectors $X_i = \{x_{i1}, x_{i2}, \ldots x_{ik}\}$, $(i = 1, 2, \ldots, T, j = 1, 2, ..., k)$, T is the size of training data vectors and k is the vector dimension. Compute the central residual vector

$$r^c = \frac{\sum_{i=1}^{T} r_i}{T}.$$

Step 1. Initialization: Select N_1 mean code scalars M_i and N_2 residual code vectors R_j for every individual of the population using random number generator, $(i = 1, 2, \ldots, N_1$ and $j = 1, 2, \ldots, N_2)$, N_1 is the mean codebook size and N_2 is the residual codebook size. P sets of N_1 mean code scalars and N_2 residual code vectors are generated, where P is the population size.

Step 2. Update: The GLA algorithm is used to update N_1 mean code scalars and N_2 residual code vectors for $MaxIte$ times for individuals in the population.

Step 3. Evaluation: The peak signal to noise ratio ($PSNR$) is used as the fitness to evaluate every individual of the population, where

$$PSNR = 10\ log \frac{\sum_{i=1}^{T} (peak\ signal)^2}{\sum_{i=1}^{T} D(X_i, \tilde{X}_i)},$$

T, X_i and $\tilde{X}_i$ are the total number of data vectors, data vector and recovered data vector, respectively.

Step 4. Selection: The survival rate P_s is used to decide whether the individual is a survivor of the current population. If a random generation number is smaller than rate P_s, then this individual survives; otherwise it does not survive.

Step 5. Crossover: The mean code scalars and residual code vectors are sorted in decreasing order according to the value of the mean code scalar and the squared Euclidean distance between the central residual vector and the residual code vector of current population. The single point crossover can be used to crossover the mean code scalars and residual code vectors separately to produce the next generation from the selected survivors in the previous step.

Step 6. Mutation: The mutation rate P_m is used to select the individual for mutation. $P_m \times (\textit{population size}) \times (\textit{codebook size})$ individuals are mutated by adding the random value to the mean code scalar and residual codevector, where the random value can be set between $\pm(\alpha^{Gen} \times \frac{\textit{deviation of the gene}}{2})$, gene is the component of the mean code scalar or residual code vector, Gen is the number of generations and $0 \leq \alpha \leq 1$.

Step 7. Termination: If the maximum number of generations has been reached or the quality is satisfied, then termination the program; otherwise, go to step 2.

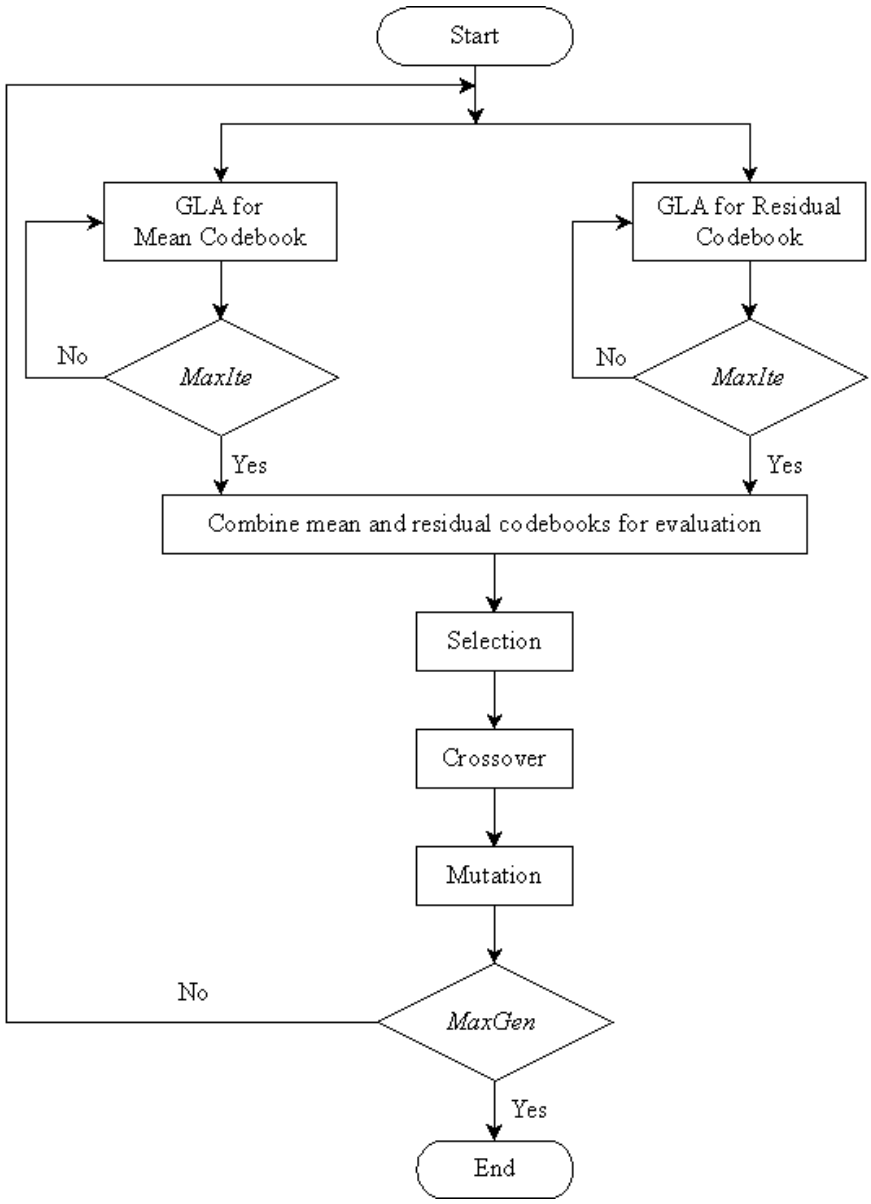


Figure 4.12: Flowchart of Genetic Clustering Algorithm for (M/R VQ)

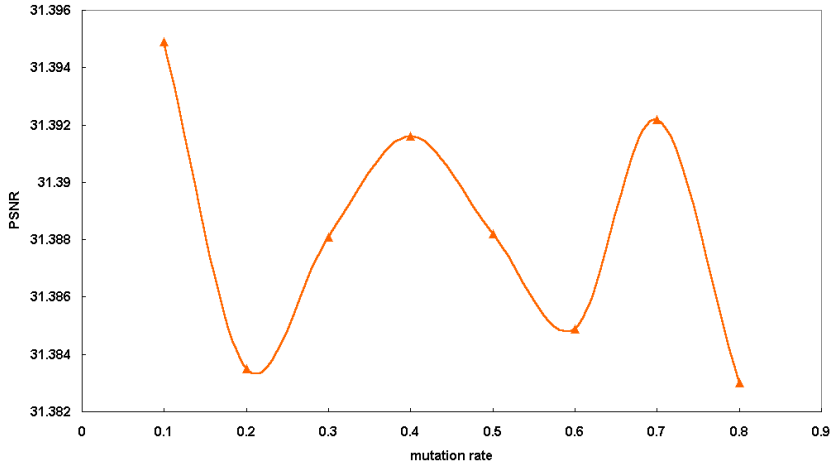


Figure 4.13. Experiment for Mutation Rate

4.3.2 Experiments

The test materials for these experiments consisted of two gray-level images, i.e., *Lena* and *Pepper* images with resolution 512×512 pixels, 8 bits per pixel. 16384 data vectors with 16 dimensions are generated from each of the images. The first experiment is to test the performance of the mutation rate, survival rate, population size, maximum number of generations Max-Gen, and the number of iterations for $GLA\ MaxIte$. Peppers image is used as the test image and mean scalars with size 32 and residual code vectors with size 32 are generated using genetic clustering algorithm. The results were averaged from 5 runs. For the experiment of mutation rate, the parameters of the population size, maximum number of generations Max-Gen, the number of iterations for $GLA\ MaxIte$, the survival rate P_s were set to be 20, 50, 5, and 0.5, respectively. The mutation rate is set from 0.1 to 0.8 with interval 0.1. As shown in Figure 4.13, the proposed algorithm is robust to the mutation rate.

For the experiment of survival rate, the parameters of the population size, maximum number of generations Max-Gen, the number of iterations for $GLA\ MaxIte$, and the mutation rate P_m were set to be 20, 50, 5, and 0.1, respectively. The survival rate is also set from 0.1 to 0.8 with interval 0.1. The experimental

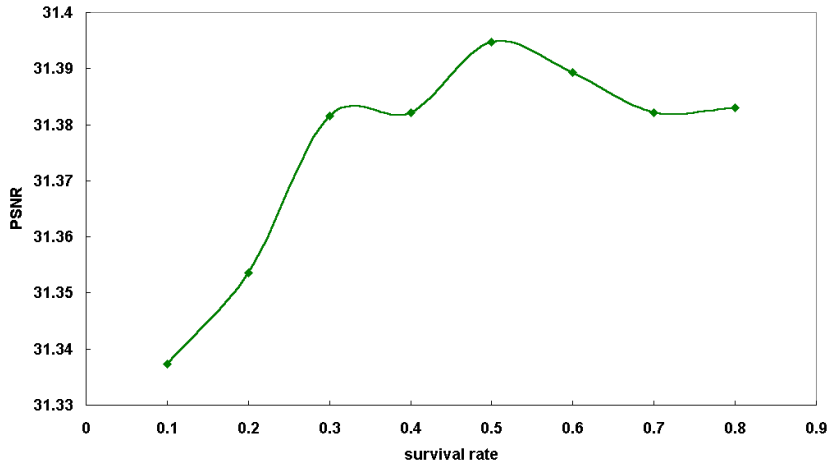


Figure 4.14. Experiment for Survival Rate

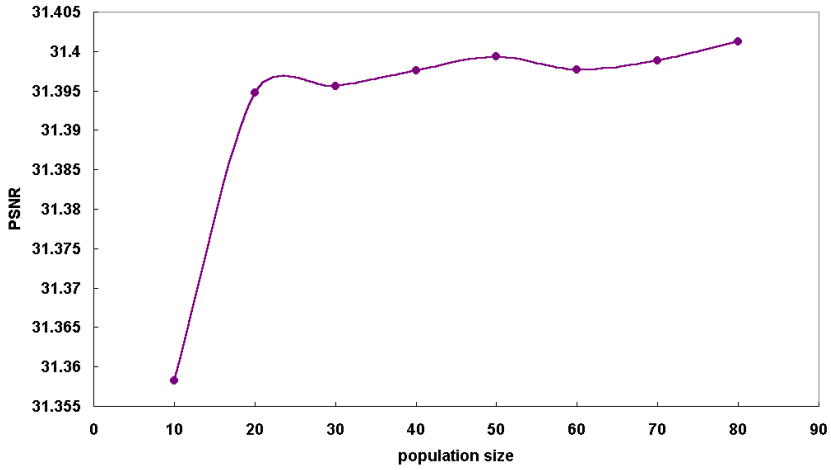


Figure 4.15. Experiment for Population Size

result is shown in Figure 4.14. The performance is similar if the survival rate is set more than 0.2.

For the experiment of population size, the parameters of the maximum number of generations Max-Gen, the number of iterations for GLA $MaxIte$, the survival rate P_s and the mutation rate P_m were set to be 50, 5, 0.5 and 0.1, respectively. The population size is set from 10 to 80 with interval 10. As shown in Figure 4.15, the performance is similar if the population size is set more than 10.

For the experiment of maximum number of generations Max-Gen, the pa-

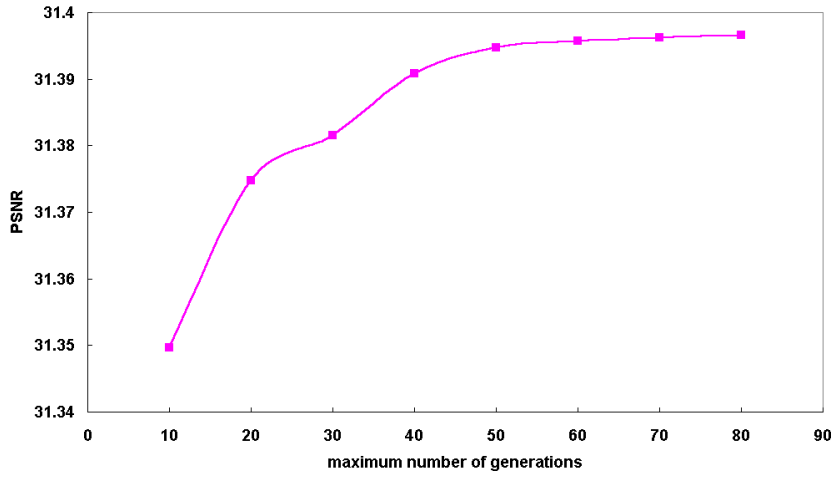


Figure 4.16. Experiment for Maximum Number of Generation

rameters of the population size, the number of iterations for GLA $MaxIte$, the survival rate P_s and the mutation rate P_m were set to be 20, 5, 0.5 and 0.1, respectively. The parameter Max-Gen is set from 10 to 80 with interval 10. As shown in Figure 4.16, the performance is only influenced a little if the maximum number of generations is more than 10.

The experiment for the number of iterations for GLA $MaxIte$, the parameters of the population size, maximum number of generations $Max-Gen$, the survival rate P_s and the mutation rate P_m were set to be 20, 50, 0.5 and 0.1, respectively. The parameter $MaxIte$ is set from 1 to 8 with interval 1. The experimental result is shown in Figure 4.17. The performance is similar if the parameter $MaxIte$ is set more than 2.

The second experiment was carried out to test the peak signal to noise ratio of the proposed genetic clustering algorithm and the conventional GLA algorithm for mean-residual vector quantization for 10 runs. The parameter ϵ for GLA algorithm was set to 0.001. The parameters of the population size, maximum number of generations Max-Gen, the number of iterations for GLA $MaxIte$, the survival rate P_s and the mutation rate P_m were set to be 20, 50, 5, 0.5 and 0.1, respectively. Firstly, the experiments compared the performance of the genetic clustering algorithm with the GLA algorithm for the mean-residual vector quan-

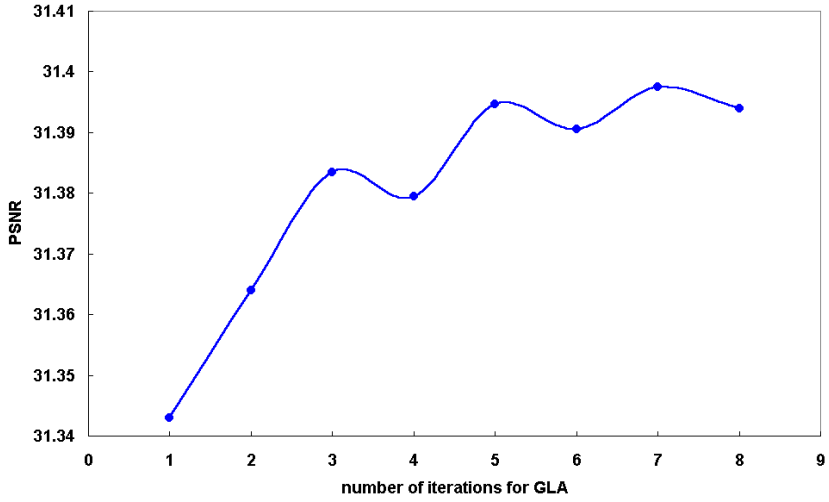


Figure 4.17. Experiment for the Number of Iteration for *GLA*

tization with mean code size 64 and residual code size 64 for Peppers and *Lena* images. As shown in Table 4.4 and Table 4.5, the proposed genetic clustering algorithm improves the $PSNR$ by 0.22 and 0.18 dB.

Table 4.4: Performance Comparison for *M/R VQ* with Mean Codebook Size 64 and Residual Codebook Size 64 Using *Pepper* Image

Seed	GA (dB)	GLA (dB)
1	32.5588	32.3736
2	32.5670	32.2930
3	32.5654	32.3510
4	32.5662	32.2936
5	32.5334	32.2855
6	32.5610	32.3727
7	32.5539	32.3615
8	32.5581	32.3242
9	32.5554	32.3630
10	32.5653	32.3275
Ave.	32.5585	32.3346

The experiments also compared the performance of the genetic clustering algorithm with the *GLA* algorithm for the mean-residual vector quantization with mean code size 128 and residual code size 128 for Peppers and *Lena* images.

Table 4.5: Performance Comparison for *M/R VQ* with Mean Codebook Size 64 and Residual Codebook Size 64 Using *Lena* Image

Seed	GA (dB)	GLA (dB)
1	32.7985	32.6291
2	32.8020	32.5925
3	32.8129	32.5572
4	32.8183	32.6642
5	32.8029	32.6386
6	32.7987	32.6020
7	32.7982	32.6687
8	32.8166	32.5967
9	32.8112	32.6280
10	32.8119	32.6723
Ave.	32.8071	32.6249

As shown in Table 4.6 and Table 4.7, the proposed genetic clustering algorithm improves the $PSNR$ by 0.36 and 0.33 dB.

Table 4.6: Performance Comparison for *M/R VQ* with Mean Codebook Size 128 and Residual Codebook Size 128 Using *Pepper* Image

Seed	GA (dB)	GLA (dB)
1	33.4780	33.0772
2	33.5299	33.1633
3	33.4962	33.1403
4	33.5062	33.1149
5	33.5167	33.0715
6	33.5057	33.2084
7	33.5359	33.0946
8	33.4673	33.1390
9	33.4713	33.2042
10	33.5208	33.2335
Ave.	33.5028	33.1447

For the experiments of mean code size 256 and residual code size 256 for Peppers and *Lena* images, the proposed genetic clustering algorithm improves

Table 4.7: Performance Comparison for *M/R VQ* with Mean Codebook Size 128 and Residual Codebook Size 128 Using *Lena* Image

Seed	GA (dB)	GLA (dB)
1	33.7917	33.4953
2	33.8132	33.4805
3	33.7900	33.4510
4	33.7898	33.4934
5	33.8007	33.4409
6	33.7859	33.4904
7	33.8024	33.4474
8	33.8115	33.5000
9	33.8206	33.4628
10	33.8058	33.4842
Ave.	33.8012	33.4746

the $PSNR$ by 0.61 and 0.61 dB shown in Table 4.8 and Table 4.9. Experimental results demonstrate the usefulness of the proposed genetic clustering algorithm for *M/R VQ*.

Table 4.8: Performance Comparison for *M/R VQ* with mean Codebook Size 256 and Residual Codebook Size 256 Using *Pepper* Image

Seed	GA (dB)	GLA (dB)
1	34.3471	33.7332
2	34.3842	33.7675
3	34.4132	33.7442
4	34.3103	33.8159
5	34.3240	33.6789
6	34.3824	33.7931
7	34.3406	33.7165
8	34.4045	33.7849
9	34.4379	33.8309
10	34.4055	33.7759
Ave.	34.3750	33.7641

Table 4.9: Performance Comparison for *M/R VQ* with Mean Codebook Size 256 and Residual Codebook Size 256 Using *Lena* Image

Seed	GA (dB)	GLA (dB)
1	34.7745	34.1504
2	34.7631	34.2104
3	34.7229	34.1366
4	34.7447	34.1040
5	34.7502	34.1887
6	34.7511	34.2024
7	34.7725	34.1145
8	34.7811	34.2021
9	34.7374	34.1145
10	34.7980	34.1075
Ave.	34.7596	34.1531

For the previous experiments, the training data and the test data are the same. For the final experiment, Peppers and *Lena* are used as the training images for *M/R VQ* with mean code size 64 and residual code size 64. F16, PEPPERS and *Lena* images are used as the test images. The parameter ϵ for GLA algorithm was set to 0.001. The parameters of the population size, maximum number of generations $Max - Gen$, the number of iterations for GLA $MaxIte$, the survival rate P_s and the mutation rate P_m were set to be 20, 50, 5, 0.5 and 0.1, respectively.

Experimental results are shown in Table 4.10. Even the F16 test image outside of the training images, the proposed genetic clustering algorithm may improve 0.2 dB compared with GLA algorithm for *M/R VQ* with mean code size 64 and residual code size 64.

In (Chu & Roddick 2002), the genetic algorithm is applied in combination with the GLA algorithm for the codebook design of mean-residual vector quantization. Experiments based on the $LENA$, PEPPERS and F16 images confirm that the proposed genetic clustering algorithm comparing with GLA algorithm may im-

Table 4.10: Performance Comparison for *M/R VQ* with Mean Codebook Size 64 and Residual Codebook Size 64 Using F16, *Pepper* and *Lena* Image

Test Image	GA (dB)	GLA (dB)
F16	29.7143	29.5162
Pepper	32.3492	32.1740
Lena	32.5417	32.4112

prove the peak signal to noise ratio by 0.2, 0.35 and 0.61 dB for mean/residual codebook size 64/64 and 128/128, respectively. The proposed algorithm is also robust to the wide range of parameters setting. Even the test image outside the training images, the genetic clustering algorithm may also get outstanding results.

4.4 Incremental Splitting Clustering

4.4.1 Introduction

Data clustering is a common practice in various fields of research and application development. For instance, in data mining, we might need to extract and capture hidden regularities diffused across a large database and store them as a limited number of representative entities. For codebook design in vector quantization, we require a small number of most representative vectors, i.e., the centers of clusters from potentially vast volumes of training data in order to minimize the quantization error.

Without loss of generality, data clustering can be formulated as a problem of finding N most representative entities, C_i, $i = 1 \ldots N$, from T supplied data items, X_i, $i = 1 \ldots T$. Generally $N \ll T$. The located $C_i's$ serve as centers of

clusters and are used to partition the T supplied data items into N mutually exclusive clusters, S_i, $i = 1 \ldots N$. A given data point X_i is considered to belong to cluster S_i if C_i is the "nearest" center to X_i.

Certain cost functions are required to measure the "distance" between given any data item and cluster center. In problem domain of k-dimensional vectors, the distance between data item X_i and the cluster centre C_j can be defined as:

$$D(X_i, C_j) = \sqrt{\frac{\sum_{p=1}^{k}(X_i^p - C_j^p)^2}{k}}$$

where X_i^p is the pth component of vector X_i.

Given the availability of a cost function, we can evaluate the quality of a particular clustering according to an error function defined in $Eq.$ 4.6. A lower E is an indication of good clustering, and vice versa.

$$E = \frac{\sum_{i=1}^{T} D(X_i, C_j)}{T} \tag{4.6}$$

where C_j is the center of the cluster to which X_i belongs.

It had been shown that an optimal clustering using an exhaustive search is prohibitive for any problem of a practical size. Numerous heuristic methods have thus been developed to achieve near-optimal clustering within reasonable computation constraints. Our algorithm, incremental splitting, performs an informed, guided search towards a more promising result in the solution space. Consistent improvement over other methods is an indication of the superiority of our approach. Our method incurs more computational overhead than others. However, we argue that the cost is justified with respect to the improvements obtained. Moreover, it may be possible to reduce the computational cost without sacrificing the advantages. We are now investigating the possibility of dynamic threshold for the k-means inner loop. The basic idea is to adopt higher threshold at early phase for coarse-grained refinement, and lower thresholds during later phases for fine-grained refinement. This scheme is expected to substantially reduce the overall computation without degradation in clustering quality. Section 4.4.2 will review

Cluster Initialization

DO

Cluster Splitting / Cluster Merge

CALL k-means method to refine the clustering

UNTIL N clusters is obtained

Figure 4.18. Pseudo Code for Local-Descent Methods

some of the widely used methods. Our approach is presented in section 4.4.3 while section 4.4.4 will report on some promising results we have so far obtained.

4.4.2 Related Works

An agglomerative clustering approach is a process in which each training data item is placed in its own cluster and these atomic clusters are gradually merged into larger and larger clusters until the desired objective is attained. In contrast, a divisive clustering approach starts with all training data in one cluster and subdividing these into several smaller clusters. Among wide variety of alternatives (Jain & Dubes 1988), local-descent methods prevail when complexity and computation cost is concerned. Most local-descent methods are derived from k-means method (MacQueen 1967) and bear a similar algorithmic structure, as shown in Figure 4.18.

The k-means method in itself an iterative process, performs local descent to search for better clustering. With k-means method, the error function in $Eq.$ 4.6 monotonically decreased. However, local-descent methods, such as k-means, run the risk of being trapped into local optima. The pseudo code in Figure 4.19 outlines the process of k-means method, where new centers are calculated as the average of all X_i within the same cluster. Different local-descent methods differ in the manner in which they deal with cluster initialization, splitting, and merging.

Three widely used heuristics are listed below together with brief descriptions:

DO

Calculate new center C_i **for each cluster** $S_i, i = 1 \ldots N$

Conduct Re-clustering using the new centers

UNTIL Percentage of improvement fall below threshold ϵ

Figure 4.19. Pseudo Code for K-Means Method

Random Initalization **(Jain & Dubes 1988):** N samples are randomly drawn from M given data and act as initial centers. K-means method is called only once. There is no cluster splitting / merging.

Binary Splitting **(Linde et al. 1980):** Start with a single cluster. In each iteration, every cluster C is spilt into two smaller ones C_1 and C_2, where $C_1 = C \times (1 + \delta)$, $C_2 = C \times (1 - \delta)$, δ is a perturbation factor. Therefore, the number of clusters doubles after each iteration. The k-means method is called to refine the clustering in each iteration.

Pair-wise Nearest Neighbor Merge **(Equitz 1989):** Start with T clusters, one cluster for each supplied data item. In each iteration, one pair of nearest neighbors is merged. The k-means method is called to refine the clustering in each iteration.

4.4.3 Proposed Algorithm – Incremental Splitting

The general structure discussed in the previous section can be regarded as an interleaved integration of local optimization and escaping mechanisms. The k-means method acts as a local optimizer, while different heuristics serve as mechanisms for escaping local optima. However, to some degree, all three heuristics are blind in that they do not make good use of the distribution of the supplied data to control the allocation of clusters. In our efforts to improve clustering, the intuitive strategy we have adopted here is to allocate more clusters to those

Start with a single cluster
DO
Split only the cluster having largest total error
CALL k-mean method to refine the clustering
UNTIL N clusters is obtained

Figure 4.20. Pseudo Code for Incremental Splitting Algorithm

regions having more sample data. With this in mind, we propose splitting only the cluster having the largest total error in each iteration. This new approach is termed *incremental splitting* and is shown in Figure 4.20. A cluster having a large error can have two causes: first, the cluster has too wide coverage; second, that the cluster contains too many samples. In either case, splitting the cluster will effectively reduce the total error and therefore improve the quality of clustering (Chu & Roddick 2001).

4.4.4 Experimental Results

A series of experiments was conducted to verify our idea. While the method will work with any dimensionality of data (by amending the distance functions accordingly) these were performed on a 2-D vector in order to visualize the distribution of resultant clusters under different heuristics. Figure 4.21 shows the distribution of 10000 sample data, which are randomly generated using Gaussian function. The threshold and the perturbation factor are set to 0.001 and 0.02 respectively in all experiments. Figure 4.22 reports the error and the distribution of 256 cluster centers using different heuristics. The incremental spitting heuristic outperforms all others, and the distribution of resultant clusters closely resembles the distribution of sample data.

To obtain more confidence in our conclusion, we examine the relative merit of different heuristics under different sample sizes and different number of clusters.

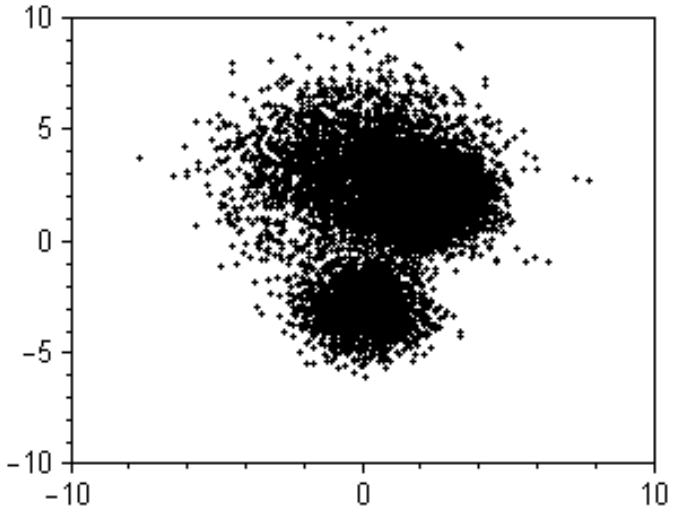


Figure 4.21. Distribution of 10,000 Sample Data

The results are summarized in Table 4.11 and Table 4.12. Random initialization and binary splitting are competitive, while our method shows consistent improvement over both methods in all cases.

From Figure 4.22, the proposed *incremental splitting clustering* may allocate more representatives (clusters) for the compact region and less representatives (clusters) for the sparse region, this will cause the reduction of the total distortion. It is typically useful for the non-uniformly distributed data and it is suitable to be applied to data compression and pattern recognition using vector quantization.

Table 4.11: Error for Different Sample Size (Number of Cluster=256)

Number of Sample Data	Random Initialization	Binary Splitting	Incremental Splitting
1000	0.153637	0.137252	0.131204
2000	0.176120	0.160780	0.155279
4000	0.184560	0.175639	0.169673
8000	0.190036	0.186481	0.179899
16000	0.194979	0.193206	0.186768
32000	0.196600	0.197891	0.192024

Table 4.12: Error for Different Number of Clusters (Sample Size = 10000)

Number of Clusters	Random Initialization	Binary Splitting	Incremental Splitting
32	0.536971	0.548424	0.535818
64	0.386860	0.391035	0.379582
128	0.272711	0.275201	0.265303
256	0.191920	0.188622	0.182477
512	0.134390	0.129469	0.1123353
1024	0.092086	0.084134	0.080145

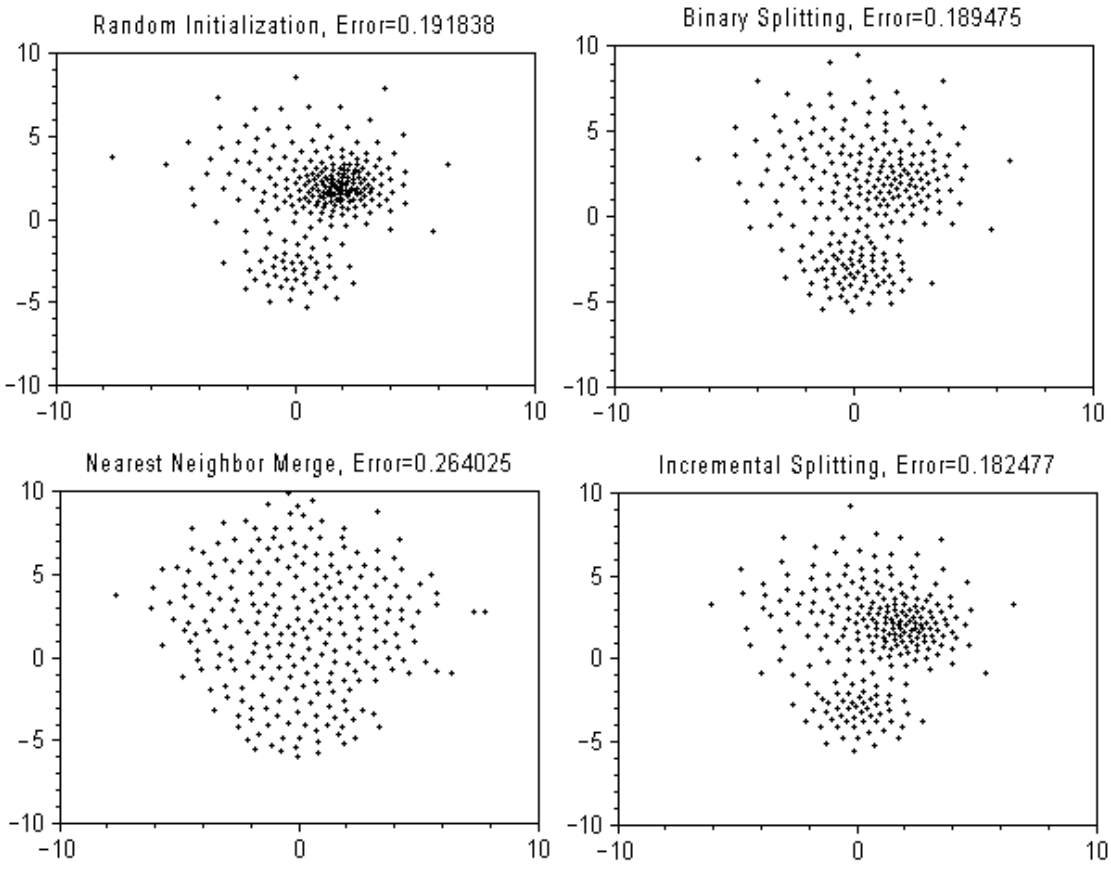


Figure 4.22. Distribution of 256 Clusters Using Different Heuristics

4.5 Labelled Bisecting K-Means Clustering Algorithm for Watermarking

A novel digital image watermarking algorithm based on labelled bisecting k-means clustering technique will be introduced in this section. The embedding process is performed by assigning the input vector to the cluster whose label is equal to the watermark bit. The security is guaranteed by two keys, the labelling key and the permutation key. In addition, the extraction process can be performed blindly. The proposed method is robust to JPEG compression and some spatial-domain processing operations. Simulation results demonstrate the effectiveness of the proposed algorithm (Chu, Roddick, Lu & Pan 2004*a*, Chu, Roddick, Lu & Pan 2003).

4.5.1 Introduction

Over the last decade, digital watermarking has been presented to complement cryptographic processes. Digital watermarking is a technique to insert a secret signal (i.e., a watermark) in digital data (namely audio, video or a digital image), which enables one to establish ownership or identify a buyer. Most of existing invisible watermarking schemes are designed for either copyright protection or content authentication. Robust watermarks (O'Ruanaidh, Dowling & Boland 1996, Cox, Kilian, Leighton & Shamoon 1997, Swanson, Bin & Tewfik 1998, Voyatzis & Pitas 1999, Pereira & Pun 2000, Wang, Doherty & Van Dyck 2002) are generally used for copyright protection and ownership verification because they are robust to nearly all kinds of image processing operations. Recently, some robust image watermarking techniques based on vector quantization (VQ) (Lu & Sun 2000, Lu et al. 2000*c*, Jo & Kim 2002, Makur & Selvi 2001, Huang, Wang & Pan 2002, Huang, Wang & Pan 2001) have been presented.

The watermark information is embedded into the encoded indices by codebook

partition or expansion technique under the constraint that the extra distortion is less than a given threshold (Lu & Sun 2000, Lu et al. 2000*c*, Jo & Kim 2002). Reference (Makur & Selvi 2001) embeds the watermark bit in the dimension information of the variable dimension reconstruction blocks of the input image. References (Huang et al. 2002, Huang, Wang & Pan 2001) embed the watermark information by utilizing the properties, such as mean and variance, of neighboring indices. In this paper, we present a novel *VQ*-based image watermarking method. In this scheme, a *VQ* codebook is first generated by the labelled bisecting k-means clustering method, where each codeword or cluster center is labelled either '0' or '1'. For each image block, the nearest codeword whose label is equal to the watermark bit is found and used to reconstruct the watermarked image. The extraction process can be performed without the original image because the embedded watermark bit is just the label of the nearest codeword for each watermarked image block.

4.5.2 Proposed Algorithm

Before describing the proposed method, we introduce some basic concepts of vector quantization. *VQ* is an efficient block-based lossy image compression technique with a high compression ratio and a simple table lookup decoder. *VQ* can be defined as a mapping from k-dimensional Euclidean space R^k into a finite codebook $C = \{c_i | i = 0, 1, \cdots, N-1\}$ where c_i is called a codeword and N is the codebook size. Before online encoding, *VQ* first generates a representative codebook off-line from a number of training vectors using the well-known *GLA* algorithm (Linde et al. 1980).

In image vector quantization, the image to be encoded is first segmented into vectors and then the encoding is operated sequentially encoded vector by vector. In the encoding stage, for each k-dimensional input vector $x = (x_1, x_2, \cdots, x_k)$, we find the nearest neighbor codeword $c_i = (c_{i1}, c_{i2}, \cdots, c_{ik})$ in the codebook

$C = \{c_0, c_1, \cdots, c_{N-1}\}$. And then the index i of the nearest neighbor codeword assigned to the input vector x is transmitted over the channel to the decoder. The decoder has the same codebook as the encoder. In the decoding phase, for each index i, the decoder merely performs a simple table look-up operation to obtain c_i and then uses c_i to reconstruct the input vector x. Compression is achieved by transmitting or storing the index of a codeword rather than the codeword itself. The main idea of the proposed *VQ*-based digital watermarking scheme is to assign each input image block to different cluster centers or codewords according to the corresponding watermark bit. Assume the codebook $C = \{c_0, c_1, \cdots, c_{N-1}\}$ has been generated by k-means algorithm.

In order to recognize the watermark bit in the extraction process without the original image, we assign a label either '0' or '1' to each cluster as well as the codeword. For description convenience, the embedding process for each input image block can be shown in Figure 4.23. Each cluster is labelled either '0' or '1' based on the labelling key generated by the labelled bisecting k-means algorithm that will be discussed later. Assume the input image block x is located in the cluster i labelled '0', while cluster j and cluster l are all labelled '1', as shown in Figure 4.23(a). If the watermark bit is equal to 0, then the codeword c_i is assigned to x. Otherwise, from all the neighboring clusters labelled '1', we find the nearest codeword c_j and assign it to x. In the extraction stage, we can easily extract the watermark bit by detecting only the label of the cluster to which the watermarked image block belongs.

Now we turn to investigate the labelling problem. People may consider adopting the random labelling method to label all of the clusters generated by k-means algorithm. However, there may be the case that all of the clusters surrounding cluster i are labelled '0' as shown in Figure 4.23(b). Thus, if the watermark bit is equal to '1', we cannot find a neighboring codeword labelled '1' to represent x. In other words, the labelling result should satisfy the following condition: Surrounding each cluster, there should be at least a cluster labelled a different label.

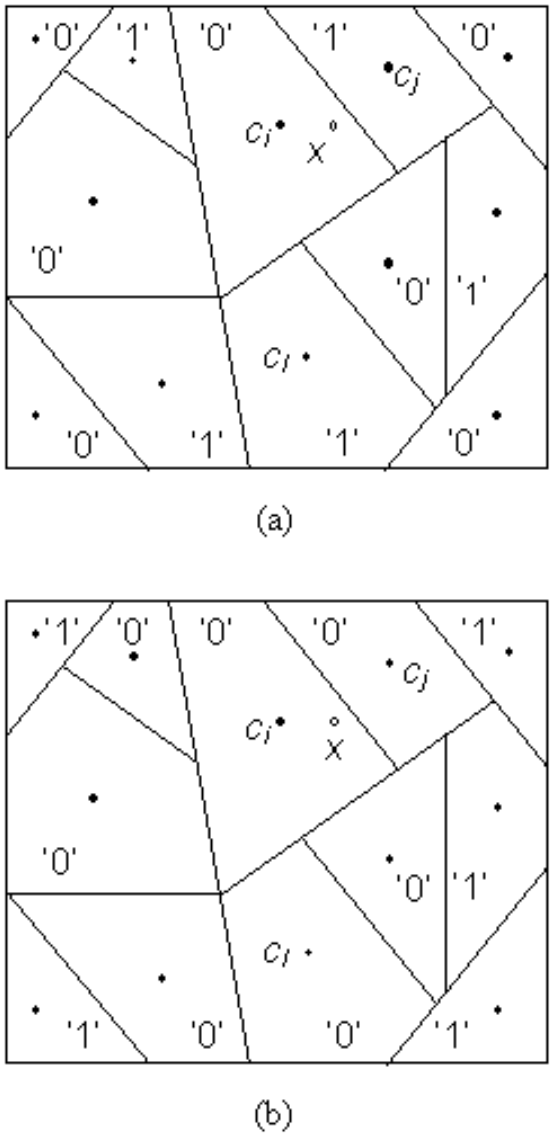


Figure 4.23: A Concrete Example to Describe The Embedding Process for Each Input Vector

In addition, to obtain a better watermarked image, the label assignment should also lead to a less extra average distortion due to the embedding operation with a random watermark bit sequence. Based on above consideration, we adopt not the conventional GLA algorithm (Linde et al. 1980) or k-means algorithm but a novel labelled bisecting k-means algorithm to generate the codeword-labelled VQ codebook. As shown in Figure 4.24, this scheme can be expressed as follows:

Step 0: The whole training set is viewed as a single cluster. Split this cluster

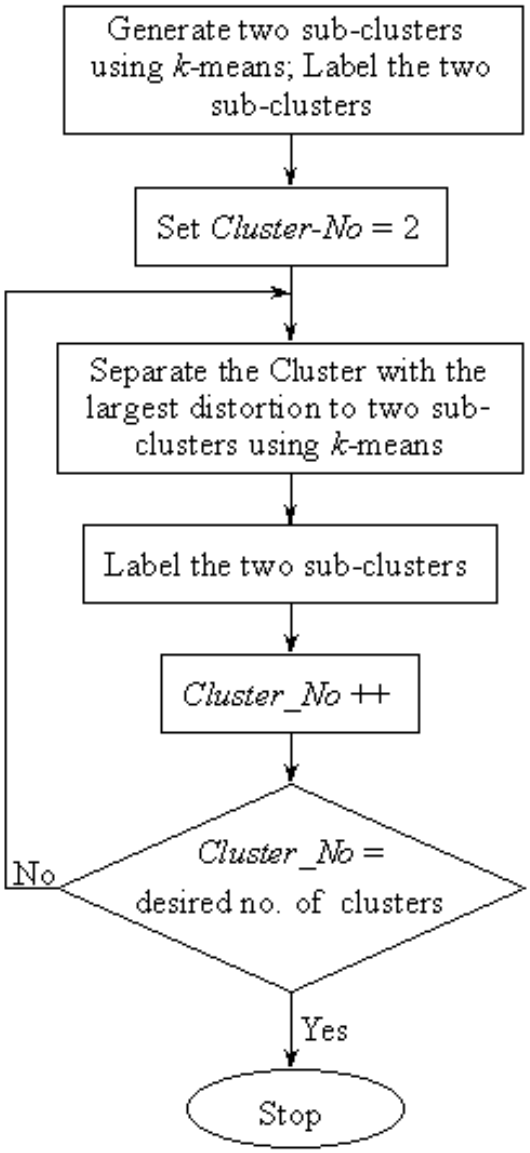


Figure 4.24: Flowchart of labelled bisecting k-means clustering algorithm

into two sub-clusters. One is labelled '0', the other is labelled '1'.

Step 1: Pick the cluster C_p that has the largest distortion to split.

Step 2: Find 2 sub-clusters using the basic k-means algorithm (Bisecting step).

Step 3: Repeat Step 2 for I_m times and take the split that produces the clustering with the highest overall similarity. Thus, we can obtain two new clusters C_a and C_b.

Step 4: For clusters C_a and C_b, find their neighboring clusters other than themselves. If C_a (or C_b) has a nearest neighboring clusters C_c labeled l, and C_b (or C_a) has no neighboring clusters, then C_a (or C_b) is labeled $1-l$ and C_b (or C_a) is labeled l. Otherwise, if C_a has a nearest neighboring clusters C_c labeled l, and C_b also has a nearest neighboring cluster C_d (may be just C_c) labeled m, then C_a is labeled $1-l$ and C_b is labeled $1-m$.

Step 5: Repeat steps 1, 2, 3 and 4 until the desired number of clusters is reached.

Step 6: Record all cluster labels and centers to form the labeling key Key_l and the final codebook C, respectively.

We can easily prove that the above labeling technique can satisfy the embedding requirement. However, this technique cannot guarantee obtaining the optimal label assignment for a random watermark bit sequence. Therefore, how to obtain the optimal label assignment is still a hard problem to be solved in the future.

After obtaining the codeword-labeled VQ codebook, we can describe the embedding process as follows. The binary watermark image is first permuted by the key Key_p to form a watermark bit sequence to be embedded, and the original image X is divided into blocks with the same size as that of the codeword. Then the embedding process can be performed block by block. For each image block x, we first find its nearest codeword c_i and compare the corresponding label p with the watermark bit w. If $p = w$, then the codeword c_i is used to reconstruct x. Otherwise, select the nearest codeword c_j from the neighboring clusters labeled w to reconstruct x. Finally, piece all reconstructed codewords together to form the watermarked image.

The extraction process is very simple and can be performed blindly. Firstly, the suspicious image is divided into blocks. Secondly, the extraction is performed block by block. For each block, find its nearest codeword and record the corresponding label according to the labeling key Key_l. Thirdly, piece all the obtained

labels together to form a bit sequence. Finally, perform the inverse permutation operation on the bit sequence to obtain the extracted watermark.

4.5.3 Experimental Results

To test the performance of the proposed method, we adopt the 256-grayscale *Lena* image of size 512×512 and a binary watermark W of size 128×128 as shown in Figure 4.25. The *Lena* image is divided into 16384 blocks of size 4×4, which are served as the training set for the labeled bisecting k-means algorithm. In our experiment, 256 to 8192 clusters are generated and labeled for embedding, and we employ the *Normalized Hamming Similarity*, NHS, to evaluate the effectiveness of the proposed algorithm. The NHS between the embedded binary watermark W and the extracted one W' is defined as

$$NHS = 1 - \frac{HD(W, W')}{number\ of\ watermark\ bits} \tag{4.7}$$

where $HD(\cdot, \cdot)$ denotes the Hamming distance between two binary strings, i.e., the number of bits different in the two binary strings. We can easily prove that $NHS \in [0, 1]$. If we acquire the higher NHS values, the embedded watermark is more similar to the extracted one. The $PSNR$ of the watermarked image is 31.11dB obtained by the proposed method for 2048 clusters. As shown in Table 4.13, the quality can be improved by the increase of the number of clusters. The NHS value of the watermark extracted from the watermarked image without any attack is equal to 1.0, which means that the proposed algorithm is able to extract the watermark perfectly because the embedded watermark and the extracted one are identical.

To check the robustness of our algorithm, we perform several attacks on the watermarked image. As shown in Table 4.14, the NHS values of the extracted watermarks for the $JPEG$ compressed watermarked images with $QF = 100\%$

Table 4.13: The relationship between the $PSNR$ of the watermarked image and the number of clusters.

Number of Clusters	32	64	128	256	512	1024	2048	4096	8192
$PSNR$ of the watermarked image (dB)	24.37	25.93	27. 63	28.76	29.80	30.63	31.11	31.99	32.47

Figure 4.25. Original Image and Watermark

and $QF = 80\%$, the VQ compressed watermarked images with Codebook 1 used in the embedding process and Codebook 2 of size 1024 trained from the *Lena* image, the median filtered and blurred watermarked images, the sharpened and contrast enhanced watermarked images and the cropped watermarked image.

From these results, we can see that the proposed algorithm is robust to $JPEG$ compression and some common spatial processing operations. In addition, the proposed algorithm can extract the watermark with $NHS = 1.0$ from the VQ-compressed watermarked image with the same codebook used in the embedding process. Figure $4.26 \sim 4.30$ show the watermarked images and the corresponding

Table 4.14. NHS **Value for Various Attacks**

Attack	NHS
No Attack	1.0
JPEG (QF=100%)	0.999
JPEG (QF=80%)	0.962
VQ (Codebook 1)	1.0
VQ (Codebook 2)	0.778
Median Filter (radius=1)	0.835
Blurring (radius=1, threshold=10)	0.872
Sharpen	0.969
Contrast Enhamement 4%	0.904
Image Cropping in The Upper-Left Corner	0.849

Figure 4.26: $JPEG$ **Compressed Watermarked Image and Corresponding Extracted watermark**

extracted watermarks for the attack of *JPEG (QF=100%)*, Median Filtering, Blurring, Sharpening and Image Cropping.

Figure 4.27: Median Filtered Watermarked Image and Corresponding Extracted Watermark

Figure 4.28: Blurred Watermarked Image and Corresponding Extracted Watermark

Figure 4.29: Sharpened Watermarked Image and Corresponding Extracted Watermark

Figure 4.30: Attacked Watermarked Image by Cropping and Corresponding Extracted Watermark

4.6 *Hadamard* Transform Based Fast Codeword Search Algorithm for High-Dimensional *VQ* Encoding

Vector quantization *(VQ)*, as we previously described in section 2.3.2, is a block-based lossy compression technique, which has been widely used in image compression and speech coding (Gersho & Gray 1992), (Linde et al. 1980). The main idea of *VQ* is to exploit the statistical dependency among vector components to reduce the spatial or temporal redundancy and obtain a high compression ratio. *VQ* can be defined as a mapping from k-dimensional Euclidean space R^k into a finite subset C of R^k. We call this finite set C the codebook and, moreover, $C = y_1, y_2, \cdots, y_N$, where y_i is a codeword and N is the codebook size. There are two key problems involved in VQ, codebook design and codeword search. The task of codebook design is to generate N most representative codewords from a large training set that consists of M training vectors, where $M \gg N$. One of the famous codebook design methods is called LBG algorithm or GLA algorithm (Linde et al. 1980). The task of codeword search is to search the best match codeword from the given codebook for the input vector. That is to say, the nearest codeword $y_j = (y_{j1}, y_{j2}, \cdots, y_{jk})$ in the codebook C is found for each input vector $x = (x_1, x_2, \cdots, x_k)$ such that the distortion between this codeword and the input vector is the smallest among all codewords.

In this section, we present an efficient nearest neighbor codeword search algorithm based on Hadamard transform for vector quantization. Four efficient elimination criteria are derived from two important inequalities based on three characteristic values in the Hadamard transform domain. Before the encoding process, all codewords in the codebook are Hadamard-transformed and sorted in the ascending order of their first elements. During the encoding process, we firstly perform the transform on the input vector and calculate its characteris-

tic values, and initialize the current closest codeword of the input vector to be the codeword whose first element of Hadamard transform is nearest to that of the input vector, and secondly use the proposed elimination criteria to find the nearest codeword to the input vector using the up-down search mechanism near the initial best-match codeword. Experimental results demonstrate the proposed algorithm is much more efficient than most existing nearest neighbor codeword search algorithms in the case of high dimension.

4.6.1 Introduction

The spatial (or temporal) inequality based algorithms eliminate unlikely codewords by utilizing the inequalities based on the characteristic values such as sum, mean, variance, and L_2 norm of the spatial vector. These inequalities can be mainly classified into five types, triangle inequalities, absolute error inequalities, mean inequalities, variance inequalities, and norm inequalities. The partial distance search *(PDS)* algorithm (Bei & Gray 1985) (see Section 2.3.3) is a simple and efficient codeword search algorithm that allows early termination of the distortion calculation between an input vector and a codeword by introducing a premature exit condition in the searching process. The triangular inequality elimination *(TIE)* criterion (see Section 2.3.4) is used in (Vidal 1986, Chen & Pan 1989, Huang & Chen 1990, Huang et al. 1992) to reject a large number of unlikely codewords. However, the TIE criterion requires considerable memory space of size $\frac{(N-1)N}{2}$ to store the distance between any pair of codewords. The *equal-average nearest neighbor search (ENNS)* algorithm (Guan & Kamel 1992, Ra & Kim 1993) uses the mean value to reject impossible codewords. This algorithm reduces a great deal of computational time compared with the conventional full search algorithm with only N additional memory. The improved algorithm, i.e., the *equal-average equal-variance nearest neighbor search (EENNS)* algorithm (Lee & Chen 1994), uses the variance as well as the mean value to reject more code-

words. This algorithm reduces the computational time further with $2N$ additional memory. The improved algorithm (Baek et al. 1997) termed *IEENNS* uses the mean and the variance of an input vector like *EENNS* but develops a new inequality between these features and the distance. Another improved *ENNS* method (Pan & Huang 1998) referred to as *IENNS*, is based on the inequality derived from the *IAEI* criterion (Pan et al. 1996*c*, Pan et al. 1996*b*). In that method, a vector is separated into two sub-vectors: one is composed of the first half of vector components and the other consists of the remaining vector components. Two inequalities based on the sums of its two sub-vectors components are used to reject those codewords that cannot be rejected by *ENNS*. Reference (Pan et al. 2003) presents so-called *sub-vector based equal-average equal-variance nearest neighbor search algorithm (SVEENNS)*, where a vector is also separated into two sub-vectors: one is composed of the first half of vector components and the other is composed of the remaining vector components. For each codeword and its two sub-vectors, the sums and variances of their vector components are computed and saved off-line. These codewords are sorted in the ascending order of the sum of their vector components. In the encoding phase, compared to $IEENNS$, four extra inequalities are used to reject those codewords which have not been rejected by $IEENNS$. Wu and Lin presented a new kick-out condition (Wu & Lin 2000) based on the norms of codewords, and we call it NOS (Norm-Ordered Search) algorithm in this paper. Recently, Lu and Sun (Lu & Sun 2003) have presented the *Equal-average Equal-variance Equal-norm Nearest Neighbor Search (EEENNS)* algorithm, which uses three significant features of a vector, mean value, variance, and norm, to reject many unlikely codewords and saves a great deal of computation time. Because the variance of a vector can be calculated from the norm and the mean of the vector, *EEENNS* algorithm can only compute and store N mean values and N norms of all codewords off-line.

The pyramid structure based algorithms reject impossible codewords by using the inequalities layer by layer. Lee and Chen (Lee & Chen 1995) present a fast

codeword search algorithm based on mean pyramids for image coding in which the vector dimension is $2^n \times 2^n$. Pan *et. al.* (Pan, Lu & Sun 2000) present a more efficient pyramid structure called the mean-variance pyramid, which can be used to reject a large number of unmatched codewords. Recently, Song and Ra (Song & Ra 2002*a*) use L_2-norm pyramid of codewords to reject unlikely codewords. The transform domain-based algorithms efficiently perform the *PDS* algorithm in wavelet or Hadamard transform, i.e., so-called *WTPDS* (Hwang et al. 1997) or *HTPDS* (Lu et al. 2000*a*) algorithm. Recently, Jiang *et. al.* (Jiang et al. 2003) also present a new Hadamard transform based *NOS* algorithm.

4.6.2 Related Existing Nearest Neighbour Codeword Search Algorithms

This section reviews some important related nearest neighbor codeword search algorithms, including *PDS*, *ENNS*, *IENNS*, *EENNS*, *IEENNS*, *SVEENNS*, *NOS* and *EEENNS* algorithms. Before describing these algorithms, we give the following definitions in advance.

Definition 1: The sum of a k-dimensional vector $x = (x_1, x_2, \cdots, x_k)$ is defined as:

$$S_x = \sum_{l=1}^{k} x_l \tag{4.8}$$

Definition 2: The mean of a k-dimensional vector $x = (x_1, x_2, \cdots, x_k)$ is defined as:

$$m_x = \frac{\sum_{l=1}^{k} x_l}{k} \tag{4.9}$$

Based on (4.8) and (4.9), we have:

$$S_x = k m_x \tag{4.10}$$

Definition 3: In the Euclidean space R^k, the *central line* l is defined as the line on which the coordinates (components) of any point (vector) have the same value. The hyperplane orthogonal to l is called an *equal-average hyperplane*.

Definition 4: The variance of a k-dimensional vector $x = (x_1, x_2, \cdots, x_k)$ is defined as:

$$v_x = \sqrt{\sum_{l=1}^{k}(x_l - m_x)^2} = \sqrt{d(x, L_x)} \tag{4.11}$$

Where $L_x = (m_x, m_x, \cdots, m_x)$ is a k-dimensional vector, which is the projection point of x on the central line l.

Definition 5: The L_2 norm of a k-dimensional vector $x = (x_1, x_2, \cdots, x_k)$ is defined as:

$$\|x\| = \sqrt{\sum_{l=1}^{k} x_l^2} \tag{4.12}$$

From above definitions, we can easily prove that the variance, the mean, and the norm of the vector x satisfy the following equation:

$$v_x^2 = \|x\|^2 - k \cdot m_x^2 \tag{4.13}$$

Definition 6: The 'so far' smallest distortion is defined as:

$$d_{min} = min\{d(x, y_i) | y_i \ has \ been \ inspected\} \tag{4.14}$$

For convenience, we often assume $d_{min} = d(x, y_p)$. In other words, we often assume the current best match codeword is y_p.

4.6.2.1 Partial Distance Search

The partial distance search *(PDS)* algorithm (Bei & Gray 1985) allows early termination of the distortion calculation between a training vector and a codeword by introducing a premature exit condition in the search process. Assume the 'so far' smallest distortion is d_{min}. If the uninspected codeword y_i satisfies the condition:

$$\sum_{l=1}^{q}(x_l - y_{il})^2 \geq d_{min} \tag{4.15}$$

which guarantees that $d(x, y_i) \geq d_{min}$, then the codeword y_i can be rejected without computing the whole distance $d(x, y_i)$, where $1 \leq q \leq k$. Although the *PDS* algorithm is not efficient enough, it can be combined with other fast search algorithms to reject the codewords that cannot be eliminated by other algorithms.

4.6.2.2 Equal-average Nearest Neighbor Search

The *ENNS* algorithm (Guan & Kamel 1992, Ra & Kim 1993) takes advantage of the fact that the nearest codeword is usually in the neighborhood of the minimum squared mean distance. The *ENNS* algorithm is based on the following Lemmata:
Lemma 1: Assume m_x and m_i are the mean values of x and y_i respectively, then

$$k \cdot (m_x - m_i)^2 \leq d(x, y_i) \tag{4.16}$$

Based on above lemma, we can easily obtain the elimination criterion described by the following theorem:
Theorem 1: Assume the 'so far' smallest distortion is d_{min}, if the mean of the uninspected codeword y_i satisfies:

$$m_i \geq m_x + \sqrt{\frac{d_{min}}{k}} \ or \ m_i \leq m_x - \sqrt{\frac{d_{min}}{k}} \tag{4.17}$$

Then y_i will not be the nearest neighbor to x.

The above theorem means that the search range can be bounded by two equal-average hyperplanes with mean values $m_{max} = m_x + \sqrt{\frac{d_{min}}{k}}$ and $m_{min} = m_x - \sqrt{\frac{d_{min}}{k}}$. In other words, the search area is bounded by two lines l_1 and l_2 perpendicular to the central line l at $L_{max} = (m_{max}, m_{max}, \cdots, m_{max})$ and $L_{min} = (m_{min}, m_{min}, \cdots, m_{min})$, respectively. To perform the *ENNS* algorithm, N mean values of all codewords should be computed off-line and stored.

In the *ENNS* algorithm, the mean of each codeword is calculated first and then these values are sorted in the increasing order. In the encoding stage, the mean of the input vector is computed, and the codeword y_p that has the absolute minimal

difference in the mean value with the input vector is selected as the tentative matching codeword. The squared Euclidean distortion d_{min} between the input vector and this tentative matching codeword is calculated. Then the codewords y_i for which $m_l \geq m_x + \sqrt{\frac{d_{min}}{k}}$ or $m_i \leq m_x - \sqrt{\frac{d_{min}}{k}}$ are eliminated. Otherwise, the *PDS* is applied to calculate the distortion and update d_{min}. The search is performed up and down near the codeword y_p iteratively. If the condition given in (4.17) is satisfied in either direction, then the search will be stopped in this direction and continued in another direction until the nearest codeword is found.

The *IENNS* algorithm (Pan & Huang 1998) is based on the Improved Absolute Erorr Inequality *(IAEI)* criterion (Pan et al. 1996*c*, Pan et al. 1996*b*). The basic idea of *IENNS* algorithm is to split each vector into two equal-sized sub-vectors to derive another two elimination criteria. Assume the vector x is split into sub-vectors x_f and x_s, and the codeword y_i is split into sub-vectors y_{if} and y_{is}, then the two extra elimination criteria can be expressed as follows:

Theorem 2: Assume S_{xf}, S_{if}, S_{xs}, and S_{is}, are the sum values of x_f, y_{if}, x_s, and y_{is} respectively, if

$$(S_{xf} - S_{if})^2 \geq \frac{k}{2} \cdot d_{min} \tag{4.18}$$

or

$$(S_{xs} - S_{is})^2 \geq \frac{k}{2} \cdot d_{min} \tag{4.19}$$

then $d(x, y_i) \geq d_{min}$, i.e., the codeword y_i can be rejected. Thus, besides the elimination criterion of *ENNS*, inequalities (4.18) and (4.19) can be used to eliminate more unlikely codewords.

4.6.2.3 Equal-average Equal-variance Nearest Neighbor Search

As discussed previously, the *ENNS* algorithm uses mean value as a feature to reject unlikely codewords. However, two vectors with the same mean value may have a large distance. Based on this condition, the *EENNS* algorithm (Lee & Chen 1994) introduces another significant feature of a vector, the variance, to

reject more codewords. First, we give another lemma for *EENNS* algorithm.

Lemma 2: Assume v_x and v_i are the variances of x and y_i respectively, then

$$(v_x - v_i)^2 \leq d(x, y_i) \tag{4.20}$$

Based on this lemma, we can obtain the following elimination criterion for *EENNS* algorithm besides the *ENNS* elimination criterion.

Theorem 3: Assume the 'so far' smallest distortion is d_{min}, if the variance of the uninspected codeword y_i satisfies

$$v_i \geq v_x + \sqrt{d_{min}} \text{ or } v_i \leq v_x - \sqrt{d_{min}} \tag{4.21}$$

Then y_i will not be the nearest neighbor to x. Based on above elimination criteria, the elimination process of the *EENNS* algorithm consists of two steps. In the first step, if $m_i \geq m_x + \sqrt{\frac{d_{min}}{k}}$ or $m_i \leq m_x - \sqrt{\frac{d_{min}}{k}}$, then the codeword y_i can be rejected. Otherwise, in the second step, if $v_i \geq v_x + \sqrt{d_{min}}$ or $v_i \leq v_x - \sqrt{d_{min}}$, then the codeword y_i can also be rejected. To perform the *EENNS* algorithm, N mean values and N variances of all codewords should be computed off-line and stored.

The *IEENNS* algorithm (Baek et al. 1997) is similar to the *EENNS* algorithm. Both algorithms use two characteristic values of a vector, the mean and the variance. However, the *IEENNS* algorithm uses the variance and the mean of a vector simultaneously, whereas the *EENNS* algorithm uses them separately. In order to use the variance and mean of a vector simultaneously, the following lemma was presented in (Baek et al. 1997):

Lemma 3:

$$k(m_x - m_i)^2 + (v_x - v_i)^2 \leq d(x, y_i) \tag{4.22}$$

Comparing *Lemmata* 1 and 2 with *Lemma* 3, we can easily see that both Lemmata 1 and 2 are the special cases of *Lemma* 3. Based on *Lemma* 3, the following elimination criterion can be obtained:

Theorem 4: Assume the 'so far' smallest distortion is d_{min}, if the mean and variance of the uninspected codeword y_i satisfy:

$$k \cdot (m_x - m_i)^2 + (v_x - v_i)^2 \geq d_{min} \tag{4.23}$$

then $d(x, y_i) \geq d_{min}$, i.e., y_i can be rejected.

Therefore, the *IEENNS* algorithm consists of two steps. The first step is the same as the *EENNS* algorithm, and the second is to use (4.23) to reject the codewords that cannot be rejected by the first step. In order to further improve the elimination efficiency, reference (Pan et al. 2003) presents a sub-vector based equal-average equal-variance nearest neighbor search algorithm *(SVEENNS)*, which exploits another two elimination criteria as follows:

Theorem 5: Assume S_{xf}, S_{if}, S_{xs}, and S_{is} are the sum values of x_f ,y_{if}, x_s, and y_{is} respectively, and v_{xf}, v_{if}, v_{xs}, and v_{is} are the variances of x_f, y_{if}, x_s, and y_{is} respectively, if

$$(S_{xf} - S_{if})^2 + \frac{k \cdot (v_{xf} - v_{if})^2}{2} \geq \frac{k \cdot d_{min}}{2} \tag{4.24}$$

or

$$(S_{xs} - S_{is})^2 + \frac{k \cdot (v_{xs} - v_{is})^2}{2} \geq \frac{k \cdot d_{min}}{2} \tag{4.25}$$

then $D(x, y_i) \geq d_{min}$, i.e., the codeword y_i can be rejected.

4.6.2.4 Norm Ordered Search

Above algorithms use mean (or sum) and/or variance of the vector to reject unlikely codewords. Recently, reference (Wu & Lin 2000) presents a kick-out condition based on norm, which is called norm-ordered search *(NOS)* algorithm in this paper. First, we can rewrite the distortion measure as

$$d(x, y_i) = \|x\|^2 + \|y_i\|^2 - 2\sum_{l=1}^{k} x_l y_{il} \tag{4.26}$$

Because $\|x\|^2$ is a common term in the distortion measure (4.26) from x to every codeword, the goal to find the codeword y_i that minimizes (4.26) is equivalent to

the goal to find the codeword that minimizes

$$d_1(x, y_i) = d(x, y_i) - \|x\|^2 = \|y_i\|^2 - 2\sum_{l=1}^{k} x_l y_{il} \tag{4.27}$$

Assume that the 'so far' nearest codeword is y_p, and the corresponding d_1-distortion is:

$$d_{1min} = d_1(x, y_p) = min\{d_1(x, y_i)|y_i \ has \ been \ inspected\} \tag{4.28}$$

According to Cauchy-Schwarz inequality, the following inequality is always true.

$$d_1(x, y_i) \geq \|y_i\|^2 - 2\|x\| \cdot \|y_i\| = \|y_i\|(\|y_i\| - 2\|x\|) \tag{4.29}$$

As a result, we can obtain the following theorem:

Theorem 6: Assume the 'so far' smallest d_1-distortion is d_{1min}, if the uninspected codeword y_i satisfies

$$\|y_i\|(\|y_i\| - 2\|x\|) \geq d_{1min} \tag{4.30}$$

then $d_1(x, y_i) \geq d_{1min}$ is guaranteed, and hence, y_i can be kicked out.

Let $f(t) = t(t-2\|x\|) = (t-\|x\|)^2 - \|x\|^2$ be a function of t, we can see that this parabola has the absolute minimum at $t = \|x\|$. In addition, $f(t)$ is monotonously increasing in the interval $(\|x\|, +\infty)$, and monotonously decreasing in the interval $(-\infty, \|x\|)$. Note that the norms $\{\|y_i\|\}_{i=1}^{N}$ can be calculated off-line and the codebook can be sorted, so that $\|y_1\| \leq \|y_2\| \leq \cdots \leq \|y_N\|$. Therefore, if the uninspected codeword y_i satisfies (4.30) and $\|y_i\| \geq \|x\|$, then all codewords y_l whose $l \geq i$ can be kicked out. On the other hand, if the uninspected codeword y_i satisfies (4.30) and $\|y_i\| \leq \|x\|$, then all codewords y_l whose $l \leq i$ can also be kicked out.

4.6.2.5 Equal-average Equal-variance Equal-norm Nearest Neighbor Search

In this section, we will discuss the *EEENNS* algorithm (Lu & Sun 2003), which uses the mean, the variance and the norm of a vector as three significant features

to speed up the closest codeword search process. According to the Cauchy-Schwarz inequality, we can easily obtain the inequality from the following Lemma:

Lemma 4: Assume $\|x\|$ and $\|y_i\|$ are the norms of x and y_i respectively, then

$$(\|x\| - \|y_i\|)^2 \leq d(x, y_i) \tag{4.31}$$

Based on Lemma 4, we can easily derive the elimination criterion denoted by the following theorem.

Theorem 7: Assume the 'so far' smallest distortion is d_{min}, if the norm $\|y_i\|$ of the uninspected codeword y_i satisfies:

$$\|y_i\| \geq \|x\| + \sqrt{d_{min}} \text{ or } \|y_i\| \leq \|x\| - \sqrt{d_{min}} \tag{4.32}$$

Then y_i will not be the nearest neighbor to x. The elimination process of the *EEENNS* algorithm consists of three steps. In the first step, if $m_i \geq m_x + \sqrt{\frac{d_{min}}{k}}$ or $m_i \leq m_x - \sqrt{\frac{d_{min}}{k}}$, then the codeword y_i can be rejected. Or else, in the second step, $v_i \geq v_x + \sqrt{d_{min}}$ or $v_i \leq v_x - \sqrt{d_{min}}$, then the codeword y_i can be rejected. Otherwise, in the third step, if $\|y_i\| \geq \|x\| + \sqrt{d_{min}}$ or $\|y_i\| \leq \|x\| - \sqrt{d_{min}}$, then the codeword y_i can be rejected. In addition, it can be easily shown that the variance v_x, the mean m_x, and the norm $\|x\|$ of the vector x satisfy (4.13). Thus, only N mean values and N norms of all codewords should be computed off-line and stored before performing the $EEENNS$ algorithm.

4.6.3 Basic Definitions And Properties

In previous section, we have reviewed several important codeword search algorithms based on spatial domain. In this section, we give some basic definitions and properties in Hadamard transform domain, which are the basis of *HTPDS* (Lu et al. 2000*a*), *TNOS* (Jiang et al. 2003) and the proposed algorithms. Let H_n be the $2^n \times 2^n$ Hadamard square matrix with elements in the set {1,-1}. By assuming all of the following vectors are k-dimensional vectors and $k = 2^n$ $(n > 0)$, the following basic definitions can be introduced:

Definition 7:
$H_1 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$ and $H_{n+1} = \begin{bmatrix} H_n & H_n \\ H_n & -H_n \end{bmatrix}$.
Definition 8: The Hadamard-transformed vector X of the vector x is defined as:

$$X = H_n x \tag{4.33}$$

Definition 9: The Hadamard-transformed variance of vector X can be defined as:

$$V_X = \sqrt{\sum_{l=2}^{k} X_l^2} \tag{4.34}$$

Definition 10: The Hadamard-transformed norm of vector X can be defined as:

$$\|X\| = \sqrt{\sum_{l=1}^{k} X_l^2} \tag{4.35}$$

Note that compared with (4.34), Equation (4.35) takes the first element of the vector X into account. Based on above definitions, we can get the following lemmata.

Lemma 5: The distortion between two spatial vectors and the distortion between the corresponding transformed vectors have the following relationship:

$$d(X, Y_i) = kd(x, y_i) \tag{4.36}$$

Proof:

$$\begin{aligned} d(X, Y_i) &= \sum_{l=1}^{k} (X_l - Y_{il})^2 = (X - Y_i)^T (X - Y_i) \\ &= [H_n(x - y_i)]^T [H_n(x - y_i)] = (x - y_i)^T H_n^T H_n (x - y_i) \\ &= (x - y_i)^T k I_k (x - y_i) = k(x - y_i)^T (x - y_i) \\ &= k \sum_{l=1}^{k} (x_l - y_{il})^2 = kd(x, y_i) \end{aligned}$$

Where I_k is the unit identity matrix of order k. This completes the proof.

Lemma 6: The first element of X is equal to the sum of all components of x, i.e.,

$$X_1 = S_x \tag{4.37}$$

Where S_x denotes the sum of vector x. This equation can be derived from the fact that each element in the first row of H_n has the same value '1'.

Lemma 7: The variance of the transformed vector X and the norm of the transformed vector X have the following relationship:

$$V_X = \sqrt{\|X\|^2 - X_1^2} \tag{4.38}$$

Based on (4.34) and (4.35), we can easily obtain (4.38).

Lemma 8: The variance of the transformed vector X and the variance of the spatial vector x have the following relationship:

$$V_X = \sqrt{k} \cdot v_x \tag{4.39}$$

According to above definitions and lemmas, we can obtain the following four properties:

Theorem 8: Assume X_1 and Y_{il} are the first elements of x and y_i respectively, and V_X and V_i are the Hadamard-transformed variances of X and Y_i respectively, then

$$(X_1 - Y_{il})^2 + (V_X - V_i)^2 \leq \sqrt{d(X, Y_i)} \tag{4.40}$$

Proof: This inequality is equivalent to the following inequalities:

$$\Leftrightarrow \quad X_1^2 + Y_{i1}^2 - 2X_1Y_{i1} + V_i^2 - 2V_XV_i \leq \textstyle\sum_{l=1}^{k}(X_l - Y_{il})^2$$

$$\Leftrightarrow \quad (X_1^2 + V_X^2) + (Y_{i1}^2 + V_i^2) - 2X_1Y_{i1} - 2V_XV_i \leq \textstyle\sum_{l=1}^{k}(X_l - Y_il)^2$$

$$\Leftrightarrow \quad \textstyle\sum_{l=1}^{k} X_l^2 + \sum_{l=1}^{k} Y_{il}^2 - 2X_1Y_{i1} - 2V_XV_i \leq \sum_{l=1}^{k} X_l^2 + \sum_{l=1}^{k} Y_{il}^2 - \sum_{l=1}^{k} 2X_lY_{il}$$

$$\Leftrightarrow \quad -2V_XV_i \leq -\textstyle\sum_{l=2}^{k} 2X_lY_{il}$$

$$\Leftrightarrow \quad V_XV_i \geq \textstyle\sum_{l=2}^{k} X_lY_{il}$$

$$\Leftrightarrow \quad \sqrt{\textstyle\sum_{l=2}^{k} X_l^2 \cdot \sum_{l=2}^{k} Y_{il}^2} \geq \textstyle\sum_{l=2}^{k} X_lY_{il}$$

The last inequality is the Cauchy-Schwarz inequality. This completes the proof.

Based on *Theorem 8*, we can easily obtain the following two useful corollaries:

Corollary 1:

$$|X_1 - Y_{i1}| \leq \sqrt{d(X, Y_i)} \tag{4.41}$$

In fact, since $(X_1 - Y_{i1})^2$ is one of the summation items in $\sum_{l=1}^{k}(X_l - Y_{il})^2$, above inequality is obviously tenable. *Corollary 2:*

$$|V_X - V_i| \leq \sqrt{d(X, Y_i)} \tag{4.42}$$

Theorem 9: Assume $\|X\|$ and $\|Y_i\|$ are the norms of X and Y_i respectively, then

$$|\|X\| - \|Y_i\|| \leq \sqrt{d(X, Y_i)} \tag{4.43}$$

Proof: This inequality is equivalent to the following inequalities:

$$\begin{aligned}
&\Leftrightarrow && \|X\|^2 + \|Y_i\|^2 - 2\|X\| \cdot \|Y_i\| \leq \textstyle\sum_{l=1}^{k}(X_l - Y_{il})^2 \\
&\Leftrightarrow && \textstyle\sum_{l=1}^{k} X_l^2 + \sum_{l=1}^{k} Y_{il}^2 - 2\|X\| \cdot \|Y_i\| \leq \sum_{l=1}^{k} X_l^2 + \sum_{l=1}^{k} Y_{il}^2 - \sum_{l=1}^{k} 2X_l Y_{il} \\
&\Leftrightarrow && 2\|X\| \cdot \|Y_i\| \geq \textstyle\sum_{l=1}^{k} 2X_l Y_{il} \\
&\Leftrightarrow && \sqrt{\textstyle\sum_{l=1}^{k} X_l^2} \cdot \sqrt{\textstyle\sum_{l=1}^{k} Y_{il}^2} \geq \textstyle\sum_{l=1}^{k} X_l Y_{il}
\end{aligned}$$

The last inequality is the Cauchy-Schwarz inequality. This completes the proof.

4.6.4 Proposed Algorithm

From *Lemma 5*, we know that the codeword that is closest to the input vector in the spatial domain is also closest to the input vector in the HT domain. Therefore we can find the corresponding best codeword in the spatial domain by searching the best codeword in the HT domain. From *Definition 7*, we know that the Hadamard transform based algorithms require the vector dimension to be the power of 2, i.e., $k = 2^n$. From *Definition 8*, we can also see that no multiplication is required for the HT.

Before describing the proposed algorithm, we first review the *HTPDS* method presented in (Lu et al. 2000*a*) and the *TNOS* method presented in (Jiang et al. 2003). It is well known that the energy of codewords can be compacted into few elements by *HT*, so *PDS* can be efficiently used to reject unlikely codewords. Suppose each codeword y_i is with dimension $k = 2^n$. Assume the 'so far' smallest

transform domain distortion is D_{min}, if the first element Y_{il} of the uninspected transformed codeword Y_i is larger than $MAXSUM = X_1 + \sqrt{D_{min}}$ or less than $MINSUM = X_1 - \sqrt{D_{min}}$, then Y_i will not be the nearest codeword of X according to *Corollary 1*. Therefore, the distance calculation is necessary only for those transformed codewords whose first elements range from *MINSUM* to *MAXSUM*. To perform the *HTPDS* algorithm, N Hadamard transformed codewords for all spatial codewords should be computed off-line and stored. *TNOS* performs the Norm-Ordered Search *(NOS)* in Hadamard transform based on *Theorem 9*. The norms $\{\|Y_i\|\}_{i=1}^{N}$ can be calculated off-line and the codebook can be sorted, so that $\|Y_1\| \leq \|Y_2\| \leq \cdots \leq \|Y_N\|$. If the uninspected codeword Y_i satisfies $|\|Y_i\| - \|X\|| \geq \sqrt{D_{min}}$ and $\|Y_i\| \geq \|X\|$, then all codewords Y_l whose $l \geq i$ can be kicked out. On the other hand, if the uninspected codeword Y_i satisfies $|\|Y_i\| - \|X\|| \geq \sqrt{D_{min}}$ and $\|Y_i\| \leq \|X\|$, then all codewords Y_l whose $l \leq i$ can also be kicked out. To perform the *TNOS* algorithm, N Hadamard transformed codewords and N transformed norms for all spatial codewords should be computed off-line and stored.

From above, we can easily see that the *HTPDS* algorithm only uses one characteristic value, i.e., the sum of the spatial vector or the first element of the transformed vector, so *HTPDS* can be viewed as the equal-average (or equal-sum) nearest neighbor search algorithm in Hadamard transform domain *(HTENNS)*. We can also easily see that the *TNOS* algorithm only uses one characteristic value, i.e., the norm of the transformed vector. To further improve the search efficiency of *HTPDS* and *TNOS* algorithms, we also consider another characteristic value, i.e., Hadamard transformed variance, in the proposed algorithm.

Based on *Theorems 8 and 9*, together with *Corollaries* 1 and 2, assume the 'so far' smallest transform domain distortion is D_{min}, four elimination criteria based on transformed vector X and codeword Y_i can be stated as follows:

Theorem 10: If

$$Y_{i1} \geq X_1 + \sqrt{D_{min}} \text{ or } Y_{i1} \leq X_1 - \sqrt{D_{min}} \tag{4.44}$$

Then $d(X, Y_i) \geq D_{min}$, and thus the codeword y_i can be eliminated.

Theorem 11: If

$$V_i \geq V_X + \sqrt{D_{min}} \text{ or } V_i \leq V_X - \sqrt{D_{min}} \tag{4.45}$$

Then $d(X, Y_i)$, and thus the codeword y_i can be eliminated.

Theorem 12: If

$$(X_1 - Y_{i1})^2 + (V_X - V_i)^2 \geq \sqrt{D_{min}} \tag{4.46}$$

Then $d(X, Y_i) \geq D_{min}$, and thus the codeword y_i can be eliminated.

Theorem 13: If

$$\|Y_i\| \geq \|X\| + \sqrt{D_{min}} \text{ or } \|Y_i\| \leq \|X\| - \sqrt{D_{min}} \tag{4.47}$$

Then $d(X, Y_i) \geq D_{min}$, and thus the codeword y_i can be eliminated.

With the above elimination criteria in hand, we can turn to describe the proposed algorithm. If we only use *Theorems* 10, 11, and 13, then the proposed algorithm can be viewed as the equal-average equal-variance equal-norm nearest neighbor search method based on Hadamard transform, which can be denoted as *HTEEENNS* (Chu, Roddick, Lu & Pan 2004*b*). Because our algorithm is based on four elimination criterion, so we can denote our algorithm as the improved *HTEEENNS*, i.e., *IHTEEENNS*.

Let $d^m(X, Y_i) = \sum_{l=1}^{m} (X_l - Y_{il})^2$ denote the partial distance between X and Y_i, where $1 \leq m \leq k$, the proposed algorithm can be illustrated as follows:

During the off-line process, HT is performed on all codewords y_i to obtain transformed codewords Y_i, and then the transformed codewords Y_i are sorted in the ascending order of their first elements. The variance V_i and norm $\|Y_i\|$ of each transformed codeword Y_i are also computed and stored in the ordered transformed codebook.

During the on-line stage, the encoding process for each input vector x can be illustrated as follows:

We first perform the HT on the input vector x to obtain X and compute its variance V_X and norm $\|X\|$, and then initialize the current closest codeword of X to be Y_p, where $p = argmin_i|X_1 - Y_{i1}|$, and compute the current minimum transform domain distortion $D_{min} = d(X, Y_p)$. We then perform the codeword search process up and down as described in *ENNS* method, and set $S_{min} = X_1 - \sqrt{D_{min}}$, $S_{max} = X_1 + \sqrt{D_{min}}$, $V_{min} = V_X - \sqrt{D_{min}}$, $V_{max} = V_X + \sqrt{D_{min}}$, $N_{min} = \|X\| - \sqrt{D_{min}}$, $N_{max} = \|X\| + \sqrt{D_{min}}$. For each codeword Y_i to be searched in each direction, if $Y_{i1} \geq S_{max}$ or $Y_{i1} \leq S_{min}$, then Y_i can be rejected and the corresponding search direction can be terminated. Otherwise, if $V_i \geq V_{max}$ or $V_i \leq V_{min}$ then Y_i can also be rejected. Otherwise, we compute $A = (X_1 - Y_{i1})^2 + (V_X - V_i)^2$ and compare A with D_{min}. If $A \geq D_{min}$, then Y_i can be also rejected. Otherwise, if $\|Y_i\| \geq N_{max}$ or $\|Y_i\| \leq N_{min}$, then Y_i can be also rejected. Otherwise, we perform the following *PDS* process. Starting from $m = 2$, for each value of m, $m = 2, 3, \cdots, k$, we first evaluate $d^m(X, Y_i)$. If $d^m(X, Y_i) > D_{min}$, then Y_i can be rejected. Otherwise, we go to next value of m and repeat the same process. This *PDS* process is repeated until Y_i is rejected or m reaches k. If $m = k$, then we compare $d(X, Y_i)$ with D_{min}. If $d(X, Y_i) < D_{min}$, then D_{min} is replaced by $d(X, Y_i)$ and the current closest codeword of X is set to be Y_i, and then S_{min}, S_{max}, V_{min}, V_{max}, N_{min}, and N_{max} are also recomputed. The search process can be stopped in the down direction once $Y_{i1} \leq S_{min}$ and stopped in the up direction once $Y_{i1} \geq S_{max}$. After the best codeword of X in the transformed domain is found, the corresponding best match codeword of x in the spatial domain is also found.

4.6.5 Experimental Results

We performed experiments on a Pentium-4 (2GHz) IBM-PC using two 512×512 monochrome images 'Lena' and 'Baboo' with 256 grey scales. Four codebooks of different sizes (512 or 1024) and dimensions($8 \times 8 = 64$ or $16 \times 16 = 256$) were designed using LBG algorithm (Linde et al. 1980) with the Lena image as the training set. The two images were used to test the effectiveness of the algorithms. The proposed *HTEEENNS* and *IHTEEENNS* algorithms were compared to the FS, PDS (Bei & Gray 1985), *ENNS* (Guan & Kamel 1992), *IENNS* (Pan & Huang 1998), *EENNS* (Lee & Chen 1994), IEENNS (Baek et al. 1997), EEENNS (Lu & Sun 2003), *SVEENNS* (Pan et al. 2003), *NOS* (Wu & Lin 2000), *TNOS* (Jiang et al. 2003) and *HTPDS* (Lu et al. 2000*a*) algorithms in terms of the CPU time and the arithmetic complexity (the average number of distance calculations per input vector) for different codebook sizes and vector dimensions as shown in Table 4.15 for 'Lena' image and Table 4.16 for 'Baboo' image. Because the Lena image is in the training set, whereas the Baboo image is a high-detail image outside the training set, the encoding time of Baboo image is much longer than that of the Lena image. From Tables 4.15 and 4.16, we can see that the proposed algorithm is superior to all other algorithms for both low-detail and high-detail images, especially in the case of high dimensionality. For Lena image encoding with the codebook of size 1024, the encoding time of proposed algorithm IHTEENNS is only about 1.3 percent of the full search algorithm on average. Comparing Table 4.15 with Table 4.16, we can see that the proposed algorithms are more efficient in the case of high-detail image.

Table 4.15: Comparisons of various fast search algorithms for 'LENA' image in the training set

Codebook size	512				1024			
Performance	CPU Time(s)		Complexity		CPU Time(s)		Complexity	
Dimension	8× 8	16× 16	8× 8	16× 16	8× 8	16× 16	8× 8	16× 16
FS	9.33	8.64	512	512	17.93	17.19	1024	1024
PDS (Bei & Gray 1985)	1.69	1.87	60.57	71.39	2.73	2.45	99.62	112.25
ENNS (Guan & Kamel 1992)	0.31	0.47	16.65	27.72	0.43	0.59	24.10	34.77
IENNS (Pan & Huang 1998)	0.21	0.40	9.19	19.00	0.24	0.40	10.83	19.21
EENNS (Lee & Chen 1994)	0.24	0.42	13.03	23.95	0.33	0.51	17.71	29.04
IEENNS (Baek et al. 1997)	0.22	0.36	13.73	20.83	0.31	0.43	19.47	24.78
EEENNS (Lu & Sun 2003)	0.24	0.40	12.68	23.07	0.31	0.49	17.25	27.94
SVEENNS (Pan et al. 2003)	0.22	0.34	14.87	21.14	0.30	0.38	19.87	23.71
NOS (Wu & Lin 2000)	0.76	0.84	49.87	61.28	1.26	1.03	81.10	74.18
TNOS (Jiang et al. 2003)	0.29	0.32	11.66	16.36	0.44	0.39	15.92	19.04
HTPDS (Lu et al. 2000*a*)	0.20	0.26	10.79	15.35	0.29	0.32	15.22	18.22
Proposed HTEEENNS	0.18	0.24	9.33	14.68	0.24	0.29	12.22	16.72
Proposed IHTEEENNS	0.18	0.24	8.82	13.44	0.23	0.26	11.61	15.01

Table 4.16: Comparisons of various fast search algorithms for 'BABOO' image in the training set

Codebook size	512				1024			
Performance	CPU Time(s)		Complexity		CPU Time(s)		Complexity	
Dimension	8× 8	16× 16	8× 8	16× 16	8× 8	16× 16	8× 8	16× 16
FS	8.75	9.17	512.00	512.00	17.92	17.69	1024.00	1024.00
PDS (Bei & Gray 1985)	3.61	4.08	139.03	161.89	6.90	7.85	270.31	315.28
ENNS (Guan & Kamel 1992)	1.30	1.64	74.96	97.30	2.54	3.49	147.06	193.14
IENNS (Pan & Huang 1998)	0.87	1.57	41.50	75.38	1.31	2.44	62.73	118.89
EENNS (Lee & Chen 1994)	1.17	1.62	65.17	94.24	2.17	3.18	122.68	185.05
IEENNS (Baek et al. 1997)	0.89	1.26	54.29	73.76	1.59	2.35	98.40	138.40
EEENNS (Lu & Sun 2003)	1.11	1.54	68.78	90.03	2.09	3.02	117.27	176.57
SVEENNS (Pan et al. 2003)	0.83	1.13	54.94	70.09	1.46	2.10	98.07	130.33
NOS (Wu & Lin 2000)	2.01	2.31	138.86	169.35	3.93	4.54	272.58	331.33
TNOS (Jiang et al. 2003)	1.08	1.23	57.06	68.11	2.07	2.70	111.22	136.12
HTPDS (Lu et al. 2000*a*)	0.87	1.13	56.61	67.63	1.69	2.13	111.65	136.17
Proposed HTEEENNS	0.77	1.00	49.10	66.06	1.42	1.96	92.63	131.40
Proposed IHTEEENNS	0.65	0.85	39.36	54.80	1.15	1.61	70.77	105.22

Chapter 5

Parallel Particle Swarm Optimization

Unlike using genetic operator, particle swarm optimization is a population-based evolutionary algorithm. In contrast of evolutionary computation technique, *PSO* is motivated from the simulating social behavior such as fish schooling and bird flocking. In this chapter, a parallel version of the particle swarm optimization (*PPSO*) algorithm is presented together with three communication strategies which can be used according to the independence of the data. The experimental results demonstrate the usefulness of the proposed *PPSO* algorithm.

5.1 History of Particle Swarm Optimization

PSO is a population based evolutionary algorithm and has similarities to the general evolutionary algorithm. However, *PSO* is motivated from the simulation of social behavior which differs from the natural selection scheme of genetic algorithms (Goldberg 1989, Davis 1991, Gen & Cheng 1997). The metaphor is that of multiple collections (a swarm) of objects moving in space and thus objects are said to possess position and velocity and are influenced by the others in the

swarm. One advantage of *PSO* is that it often locates near optima significantly faster than evolutionary optimization (Angeline 1998, Eberhart & Shi 1998).

PSO processes the search scheme using populations of particles which correspond to the use of individuals in genetic algorithms. Each particle is equivalent to a candidate solution of a problem. The particle will move according to an adjusted velocity, which is based on the corresponding particles experience and the experience of its companions. For the D-dimensional function $f(.)$, the ith particle for the tth iteration can be represented as

$$X_i^t = (x_i^t(1), x_i^t(2), \ldots, x_i^t(D)). \tag{5.1}$$

Assume that the best previous position of the ith particle at the tth iteration is represented as

$$P_i^t = (p_i^t(1), p_i^t(2), \ldots, p_i^t(D)), \tag{5.2}$$

then

$$f(P_i^t) \leq f(P_i^{t-1}) \leq \ldots \leq f(P_i^1). \tag{5.3}$$

The velocity of the ith particle at the tth iteration can be represented as

$$V_i^t = (v_i^t(1), v_i^t(2), \ldots, v_i^t(D)). \tag{5.4}$$

$$G^t = (X^t(1), X^t(2), \ldots, X^t(D)) \tag{5.5}$$

is defined as the *best* position amongst all particles from the first iteration to the tth iteration, where *best* is defined by some function of the swarm.

The original particle swarm optimization algorithm can be expressed as follows:

$$V_i^{t+1} = V_i^t + C_1.r_1.(P_i^t - X_i^t) + C_2.r_2.(G^t - X_i^t) \tag{5.6}$$

$$X_i^{t+1} = X_i^t + V_i^{t+1}, i = 0, \ldots N-1 \tag{5.7}$$

where N is particle size, $-V_{max} \leq V_i^{t+1} \leq V_{max}$, ($V_{max}$ is the maximum velocity) and r_1 and r_2 are random numbers such that $0 \leq r_1, r_2 \leq 1$. A discrete

binary version of the particle swarm optimization algorithm was also proposed by Kennedy and Eberhart (Kennedy & Eberhart 1997).

The particle swarm optimization algorithm has been applied *inter alia* to optimize reactive power and voltage control (Fukuyama & Yoshida 2001) and human tremor (Eberhart & Hu 1999). A modified version of the particle swarm optimizer (Shi & Eberhart 1998) and an adaption using the inertia weight[1] of the modified particle swarm (Shi & Eberhart 1999) have also been presented. The latter version of the modified particle swarm optimizer can be expressed as

$$V_i^{t+1} = W^t.V_i^t + C_1.r_1.(P_i^t - X_i^t) + C_2.r_2.(G^t - X_i^t) \tag{5.8}$$

$$X_i^{t+1} = X_i^t + V_i^{t+1}, i = 0, \ldots N-1 \tag{5.9}$$

where W^t is the inertia weight at the tth iteration. Shi and Eberhart (Shi & Eberhart 2001) have also applied fuzzy theory to adapt the particle swarm optimization algorithm. In addition, the explosion, stability and convergence of the *PSO* has been analyzed by Clerc and Kennedy (Clerc & Kennedy 2002).

To experience the power of particle swarm optimization, applied program to the following test function, as visualized in Figure 5.1.

$$F_2(x,y) = -xsin(\sqrt{\mid x \mid}) - ysin(\sqrt{\mid y \mid}),\ -500 < x, y < 500 \tag{5.10}$$

where global optimum is at $F_2(-420.97, -420.97) = 837.97$.

In the tests above, both learning factors, c_1 and c_2, are set to a value of 2, and a variable inertia weight w is used according to the suggestion from Shi and Eberhart (1999). Figure 5.2 reports the progress of particle swarm optimization on the test function $F_2(x,y)$ for the first 300 iterations. At the end of 1000 iterations, $F_2(-420.97, -420.96) = 837.97$ is located, which is close to the global optimum.

[1]Strictly speaking, the term should be simply *inertia* but we use the term as per (Shi & Eberhart 1998).

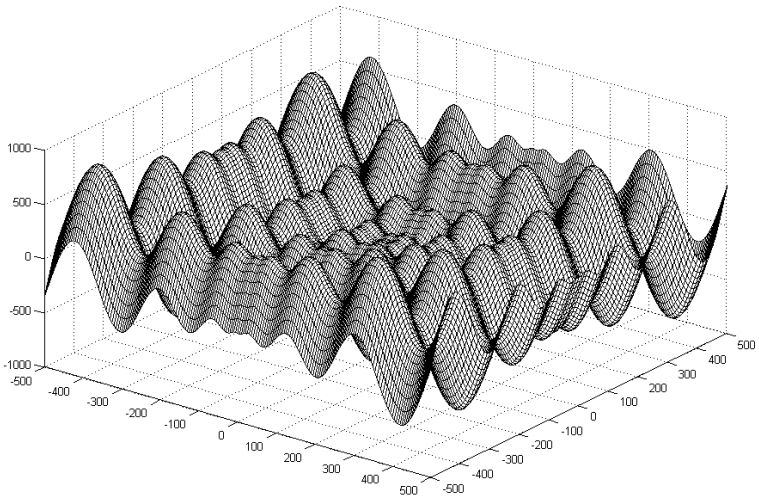


Figure 5.1. Object function F_2.

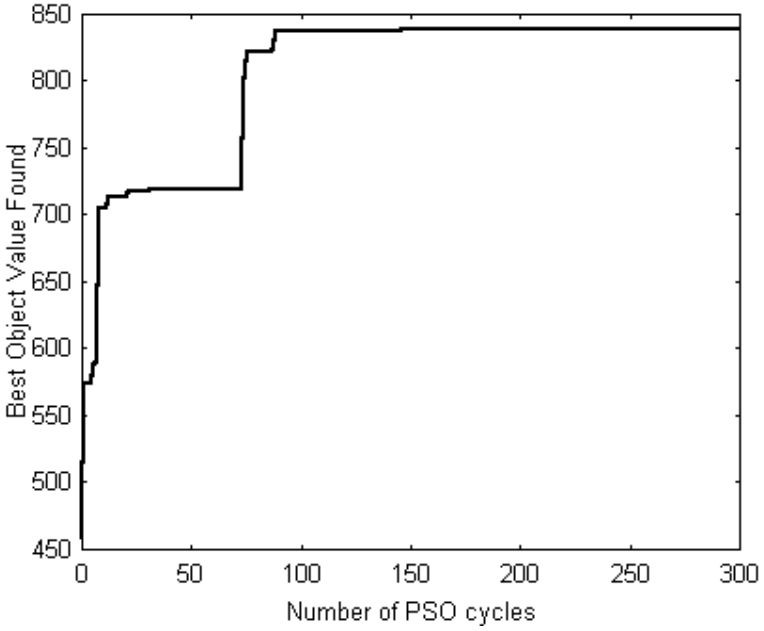
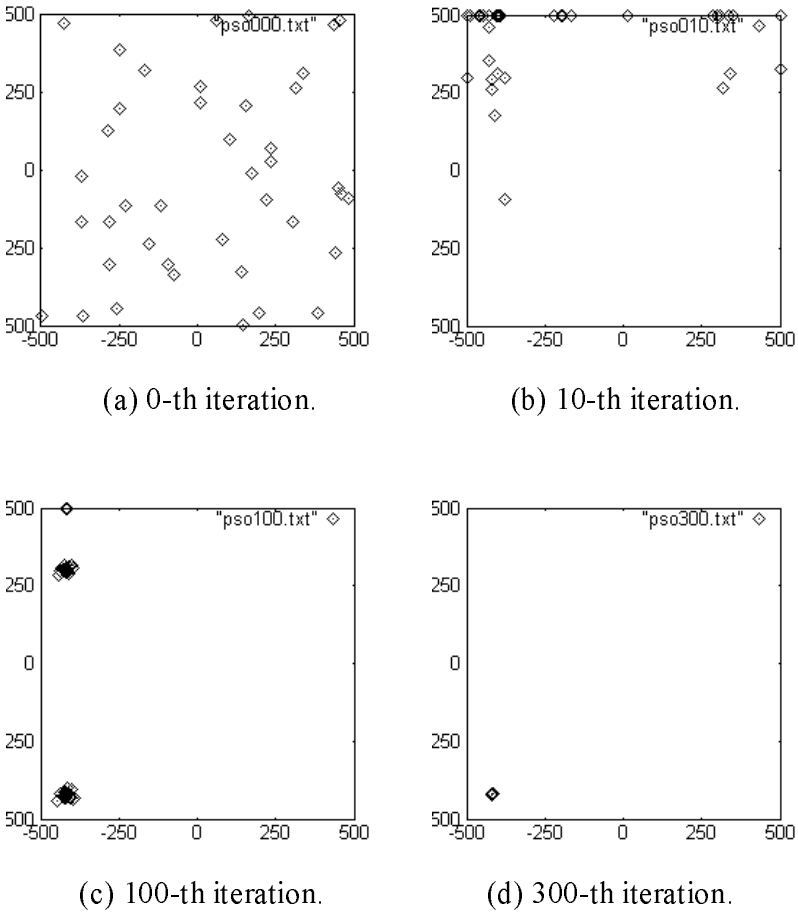


Figure 5.2. Progress of *PSO* on object function F_2.



(a) 0-th iteration. (b) 10-th iteration.

(c) 100-th iteration. (d) 300-th iteration.

Figure 5.3. The distribution of particles at different iterations.

It is worthwhile to look into the dynamics of particle swarm optimization. Figure 5.3 presents the distribution of particles at different iterations. There is a clear trend that particles start from their initial positions and fly toward the global optimum.

5.2 Particle Swarm Optimization with Communication Strategies

5.2.1 Motivation and Description

Parallel processing is concerned with producing the same results using multiple processors with the goal of reducing the running time. The two most common parallel processing methods are pipeline processing and data parallelism. The principle of pipeline processing is to separate the problem into a cascade of tasks where each of the tasks is executed by an individual processor. Data are transmitted through each processor which executes a different part of the process on each of the data elements. Since the program is distributed over the processors in the pipeline and the data moves from one processor to the next, no processor can proceed until the previous processor has finished its task. Data parallelism is an alternative approach which involves distributing the data to be processed amongst all processors which then executes the same procedure on each subset of the data. Data parallelism has been applied fairly widely including to genetic algorithms.

The *parallel genetic algorithm* (PGA) works by dividing the population into several groups and running the same algorithm over each group using different processors (Cohoon, Hegde, Martine & Richards 1987). However, the purpose of applying parallel processing to genetic algorithms goes further than merely being a hardware accelerator. Rather a distributed formulation is developed which gives better solutions with lower overall computation. In order to achieve this, a level of communication between the groups is performed every fixed number of generations. That is, the parallel genetic algorithm periodically selects promising individuals from each subpopulation and migrates them to different subpopulations. With this migration (communication), each subpopulation will receive some new and promising chromosomes to replace the poorer chromosomes in a

subpopulation. This strategy helps to avoid premature convergence. The parallel genetic algorithm has been successfully applied to vector quantization based communication via noisy channels (Pan, McInnes & Jack 1996*a*).

In this work, the spirit of the data parallelism method is utilized to create a *parallel particle swarm optimization* (*PPSO*) algorithm (Chu, Roddick & Pan 2004*b*).

It is difficult to find an algorithm which is efficient and effective for all types of problem. As shown in previous work by Shi and Eberhart (2001), the fuzzy adaptive particle swarm optimization algorithm is effective for solutions which are independent or are loosely correlated such as the generalized Rastrigrin or Rosenbrock functions. However, it is not effective when solutions are highly correlated such as for the Griewank function. Our research has indicated that the performance of the *PPSO* can be highly dependent on the level of correlation between parameters and the nature of the communication strategy – this can be explained as follows. Assuming the parameters of solutions are independent or are only loosely correlated. If we tune the value of one parameter to get a better solution cost by keeping the other parameters constant, the value of this parameter is always in the neighborhood of the best solution. Based on this observation, we may update the best particle among all particles to each group and mutate to replace the poorer particles in each group very frequently. However, the above observation is not true if the parameters of solutions are strongly correlated. In fact the best solutions can be spread throughout the search space. In this case, we need to keep the parameters be divergent and the best particles cannot be used to replace the poorer particles for all groups and the communication frequency should be infrequent. Thus in this case it is more effective to limit the communication to the neighbourhood only in order to retain the divergence. If the properties of the parameters are unknown, we may apply the communication strategy 3 which is the hybrid version of the communication strategy 1 and 2. Three communication strategies are thus presented and experiments have been

carried out which show the utility of each strategy.

The mathematical form of the parallel particle swarm optimization algorithm can be expressed as follows:

$$V_{i,j}^{t+1} = W^t.V_{i,j}^t + C_1.r_1.(P_{i,j}^t - X_{i,j}^t) + C_2.r_2.(G_j^t - X_i^t) \tag{5.11}$$

$$X_{i,j}^{t+1} = X_{i,j}^t + V_{i,j}^{t+1} \tag{5.12}$$

$$f(G^t) \leq f(G_j^t) \tag{5.13}$$

where $i = 0, \ldots N_j - 1, j = 0, \ldots S - 1$, S $(= 2^m)$ is the number of groups (and m is a positive integer), N_j is the particle size for the jth group, $X_{i,j}^t$ is the ith particle position in the jth group at the tth iteration, $V_{i,j}^t$ is the velocity of the ith particle in the jth group at the tth iteration, G_j^t is the best position among all particles of the jth group from the first iteration to the tth iteration and G^t is the best position among all particles in all groups from the first iteration to the tth iteration.

As discussed above, three communication strategies have been developed for *PPSO*. The first strategy, shown in Figure 5.4, is based on the observation that if parameters are independent or are only loosely correlated, then the better particles are likely to obtain good results quite quickly. Thus multiple copies of the best particles for all groups G^t are mutated and those mutated particles migrate and replace the poorer particles in the other groups every R_1 iterations.

However, if the parameters of a solution are loosely correlated the better particles in each group tend not to obtain optimum results particularly quickly. In this case, a second communication strategy may be applied as depicted in Figure 5.5. This strategy is based on self-adjustment in each group. The best particle in each group G_j^t is migrated to its neighbour groups to replace some of the more poorly performing particles every R_2 iterations. Since we have defined the number of clusters S as a power of two, *neighbours* are defined as being those clusters where the binary representation of the cluster number j differs by one bit.

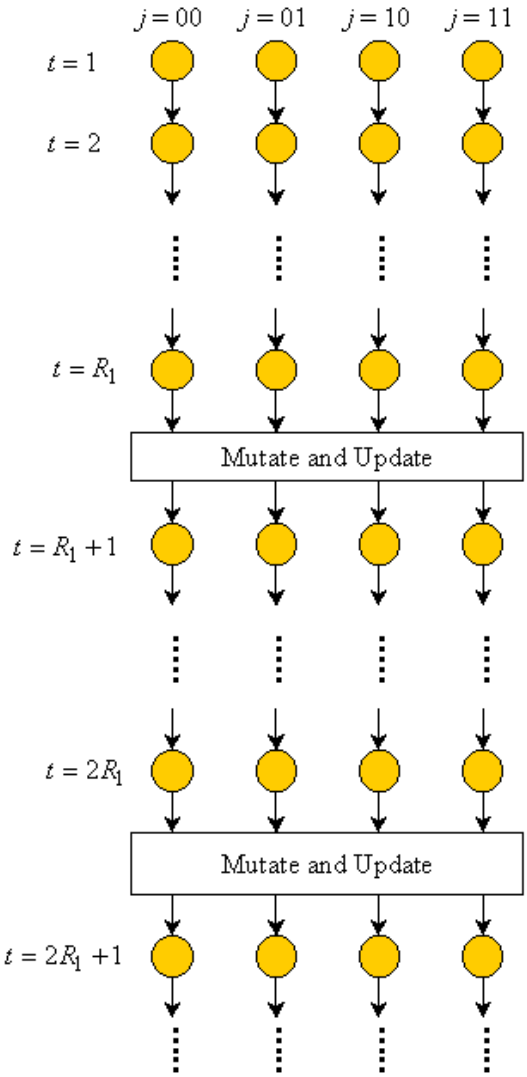


Figure 5.4: Communication Strategy for Loosely Correlated Parameters.

When the correlation property of the solution space is known the first and second communication strategies work well. However, they can perform poorly if applied in the wrong situation. As a result, in the cases where the correlation property is unknown, a hybrid communication strategy can be applied. This hybrid strategy separates the groups into two equal sized subgroups with the first subgroup applying the first strategy every R_1 iterations and all groups applying the second strategy every R_2 iterations as depicted in Figure 5.6.

The complete parallel particle swarm optimization ($PPSO$) algorithm with its

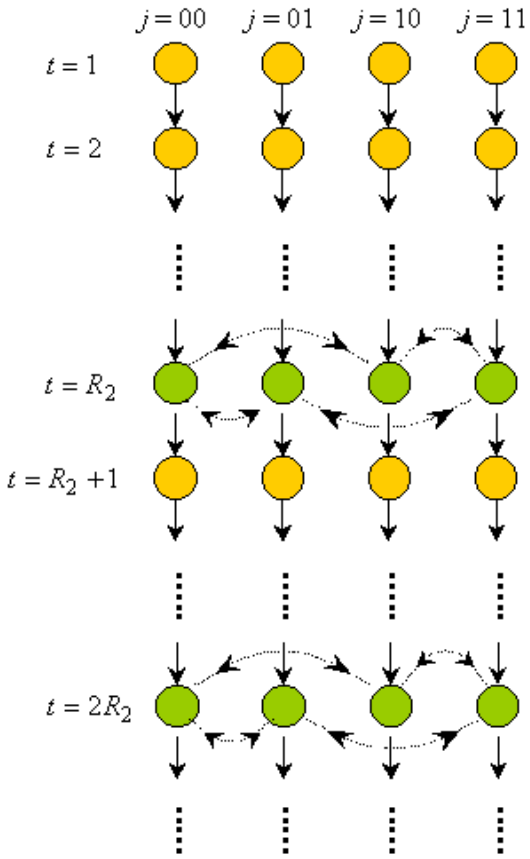


Figure 5.5: Communication Strategy for strongly correlated parameters.

three communication strategies is as follows:

1. **Initialization:** Generate N_j particles $X_{i,j}^t$ for the jth group, $i = 0, \ldots N_j - 1, j = 0, \ldots S - 1$, S is the number of groups, N_j is the particle size for the jth group and t is the iteration number. Set $t = 1$.

2. **Evaluation:** The value of $f(X_{i,j}^t)$ for every particle in each group is evaluated.

3. **Update:** Update the velocity and particle positions using equations (5.11), (5.12) and (5.13).

4. **Communication:** Three possible communication strategies are as follows:

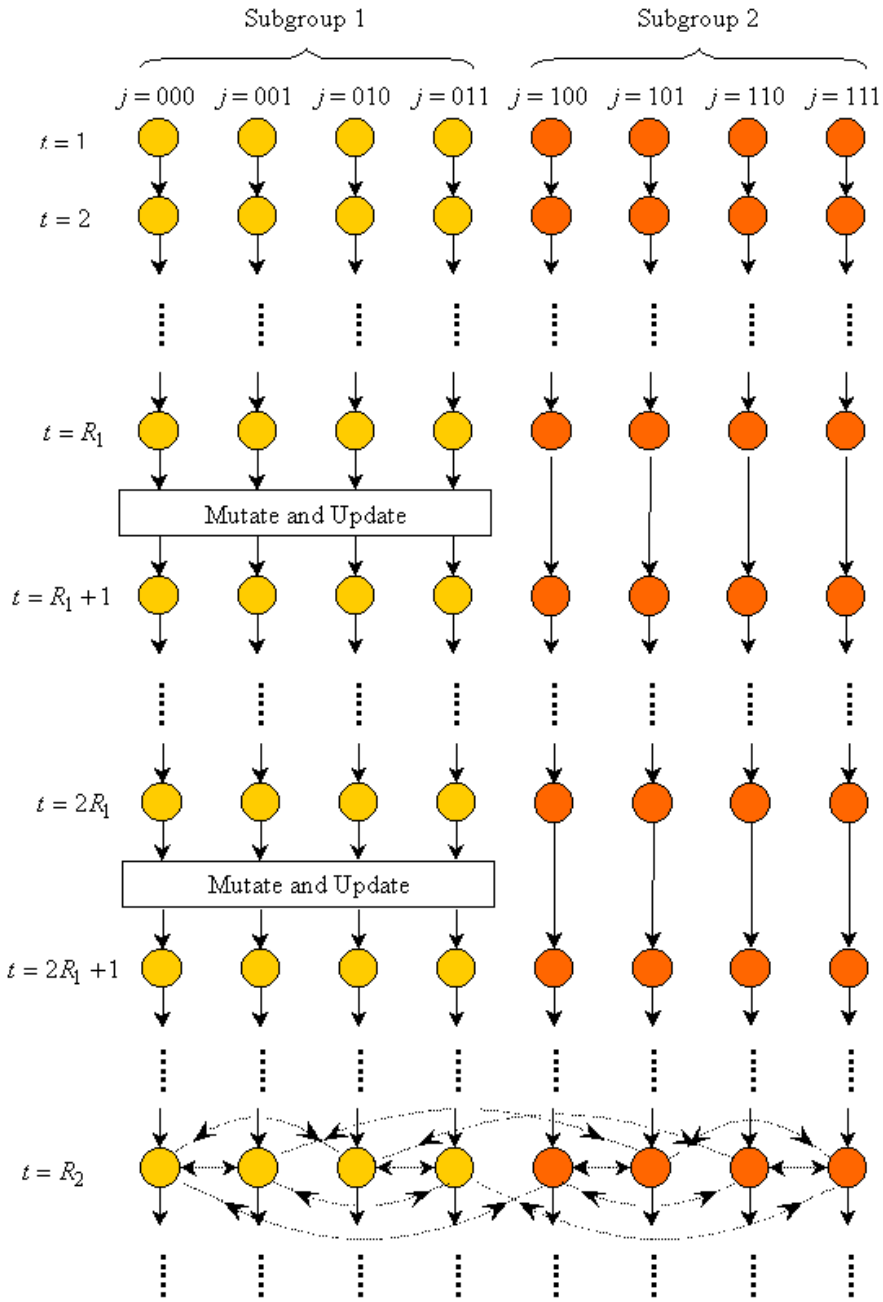


Figure 5.6: A General Communication Strategy for Unknown Correlation Between Parameters.

Strategy 1: Migrate the best particle among all particles G^t to each group and mutate G^t to replace the poorer particles in each group and update G_j^t with G^t for each group, every R_1 iterations.

Strategy 2: Migrate the best particle position G_j^t of the jth group to its

neighbouring groups to substitute for some poorer particles, every R_2 iterations.

Strategy 3: Separate the groups into two subgroups. Apply communication strategy 1 to subgroup 1 every R_1 iterations and communication strategy 2 to both subgroup 1 and subgroup 2 for every R_2 iterations.

5. **Termination:** Steps 2 to 5 are repeated until the predefined value of the function or some maximum number of iterations has been reached. Record the best value of the function $f(G^t)$ and the best particle position among all particles G^t.

5.2.2 Experiments

Let $X = \{x_1, x_2, \ldots, x_n\}$ be an n-dimensional real-value vector. The Rosenbrock function can be expressed as follows:

$$f_1(X) = \sum_{i=1}^{n} (100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2). \tag{5.14}$$

The second function used in the experiments was the generalized Rastrigrin function which can be expressed as:

$$f_2(X) = \sum_{i=1}^{n} (x_i^2 - 10cos(2\pi x_i)^2 + 10). \tag{5.15}$$

The third function used was the generalized Griewank function as follows:

$$f_3(X) = \frac{1}{400} \sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} cos\left(\frac{x_i}{\sqrt{i}}\right) + 1. \tag{5.16}$$

Experiments were carried out to test the performance of the *PPSO* communication strategies. They confirmed that the first strategy works best when the parameters of the solution are loosely correlated such as for the Rastrigrin and Rosenbrock functions while the second strategy applies when the parameters of the solution are more strongly correlated as is the case for the Griewank function.

A final experiment tested the performance of the third strategy for all three functions. All three experiments are compared with the linearly decreasing inertia weight *PSO* (Shi & Eberhart 1999) for 50 runs.

The parameters of the functions for *PSO* and *PPSO* were set as in Table 5.1. We did not limit the value of X, C_1 and C_2 were set to 2, the maximum number of iterations was 2000, $W_t^0 = 0.9$, $W_t^{2000} = 0.4$ and the number of dimensions was set to 30.

Table 5.1. Asymmetric initialization ranges and V_{max} values

Function	Asymmetric Initialization Range	V_{max}
f_1	$15 \leq x_i \leq 30$	100
f_2	$2.56 \leq x_i \leq 5.12$	10
f_3	$300 \leq x_i \leq 600$	600

To ensure a fair comparison, the number of groups $\times$ the number of particles per group was kept constant – the particle size for the *PSO* was 160, one swarm with 160 particles, as reported by 1×160. For *PPSO*, the particle size was also set to be 160 that was divided into 8 groups with 20 particles in each group (i.e. 8×20), 4 groups with 40 particles in each group (i.e. 4×40) and 2 groups with 80 particles in each group (i.e. 2×80), respectively. For the first experiment, the number of iterations for communication was set to 20 and the best particle position was migrated and mutated to substitute 25%, 50%, 75% and 100% of the poorer particles in the receiving group.

As shown in Table 5.2 and Table 5.3, the first communication strategy is effective for the parameters of solution that are independent or loosely correlated.

For the second experiment, the number of iterations for communication was set to 100 and the number of poorer particles substituted at each receiving group was set to 1 and 2. Experimental results are shown in Table 5.4 and show that the second communication strategy for 8 groups may improve the performance

Table 5.2: Performance Comparison of *PSO* and *PPSO* with The First Communication Strategy for Rosenbrock Function

Percentage of Migration	**cost of** $f_1(X)$			
	PSO	PPSO(2,80)	PPSO(4,40)	PPSO(8,20)
None	108.74			
25%		65.38	98.56	75.95
50%		75.99	61.10	67.18
75%		61.19	64.51	59.96
100%		68.44	60.20	50.88

Table 5.3: Performance Comparison of *PSO* and *PPSO* with The First Communication Strategy for Rastrigin Function

Percentage of Migration	**cost of** $f_2(X)$			
	PSO	PPSO(2,80)	PPSO(4,40)	PPSO(8,20)
None	24.54			
25%		16.88	17.91	16.06
50%		15.12	12.88	12.84
75%		12.88	11.18	11.02
100%		11.24	10.51	10.03

by up to 66%.

Finally, in the case of the third experiment, the parameters are the same as

Table 5.4: Performance Comparison of *PSO* and *PPSO* with The Second Communication Strategy for Griewank Function

Number of Migration	**cost of** $f_3(X)$			
	PSO	PPSO(2,80)	PPSO(4,40)	PPSO(8,20)
None	0.01191			
1		0.01137	0.00822	0.00404
2		0.01028	0.01004	0.00601

the first and second experiments. 50% of particles are substituted in the receiving group for the first communication strategy and 2 particles are substituted in the receiving group for the second communication strategy. As shown in Table 5.5, the hybrid communication strategy can be effective for all three functions.

Table 5.5: Performance Comparison of *PSO* and *PPSO* with The Third Communication Strategy

Function	**cost**		
	PSO	PPSO(4,40)	PPSO(8,20)
Rosenbrock	108.74	76.59	82.62
Rastrigin	24.54	18.63	19.14
Griewank	0.01191	0.01053	0.00989

Chapter 6

Parallel and Constrained Ant Colony Optimizations

Parallelization strategies for *AS* (Bullnheimer, Kotsis & Strauss 1997) and *ACS* (Stützle 1998) have been investigated, however, these studies are based on simply applying *AS* or *ACS* on the multi-processor, ie. the parallelization strategies simply share the computation load over several processors. No experiments demonstrate the sum of the computation time for all processors can be reduced compared with the single processor works on the *AS* or *ACS*.

In this chapter, we apply the concept of parallel processing to *Ant Colony System* (*ACS*) and a *Parallel Ant Colony System* (*PACS*) is proposed. The purpose of the *PAS* and *PACS* is not just to reduce the computation time. Rather a parallel formulation is developed which gives not only reduces the elapsed and the computation time but also obtains a better solution. The artificial ants are firstly generated and separated into several groups. The Ant Colony System algorithm is then applied to each group and communication between groups is applied according to some fixed cycles. The basic idea of the communication is to update the pheromone level for each route according to the best routes found by neighbouring groups or, in some cases, all groups. Three and Sever communi-

cation strategies are separately proposed for $PACS$. Experimental results based on the traveling salesman problem confirm the efficiency and effectiveness of the proposed $PACS$ (Chu, Roddick, Pan & Su 2003, Chu, Roddick & Pan 2004*a*).

6.1 History of Ant System and Ant Colony System

The Ant System algorithm (Colorni, Dorigo & Maniezzo 1991, Dorigo et al. 1996) is a cooperative population-based search algorithm inspired by the behaviour of real ants. As each ant construct a route from nest to food by stochastically following the quantities of pheromone level, the intensity of laying pheromone will bias the path-choosing decision-make of subsequent ants. It is a new member of the class of metaheuristics joining algorithms such as simulated annealing (Kirkpatrick et al. 1983, Huang, Pan, Lu, Sun & Hang 2001), genetic algorithms (Goldberg 1989, Pan et al. 1995), tabu search approaches (Glover & Laguna 1997, Pan & Chu 1996) and neural networks (Kohonen 1995, Kung 1993). In common with many of these the Ant System algorithm is similarly derived from nature.

The operation of ant systems can be illustrated by the classical *Travelling Salesman Problem* (see Figure 6.1 for example). In the TSP problem, a travelling salesman problem is looking for a route which covers all cities with minimal total distance. Suppose there are n cities and m ants. The entire algorithm starts with initial pheromone intensity set to τ_0 on all edges. In every subsequent ant system cycle, or episode, each ant begins its trip from a randomly selected starting city and is required to visit every city exactly once (a Hamiltonian Circuit). The experience gained in this phase is then used to update the pheromone intensity on all edges.

Given a finite set of cities and the distance between each pair of cities, the

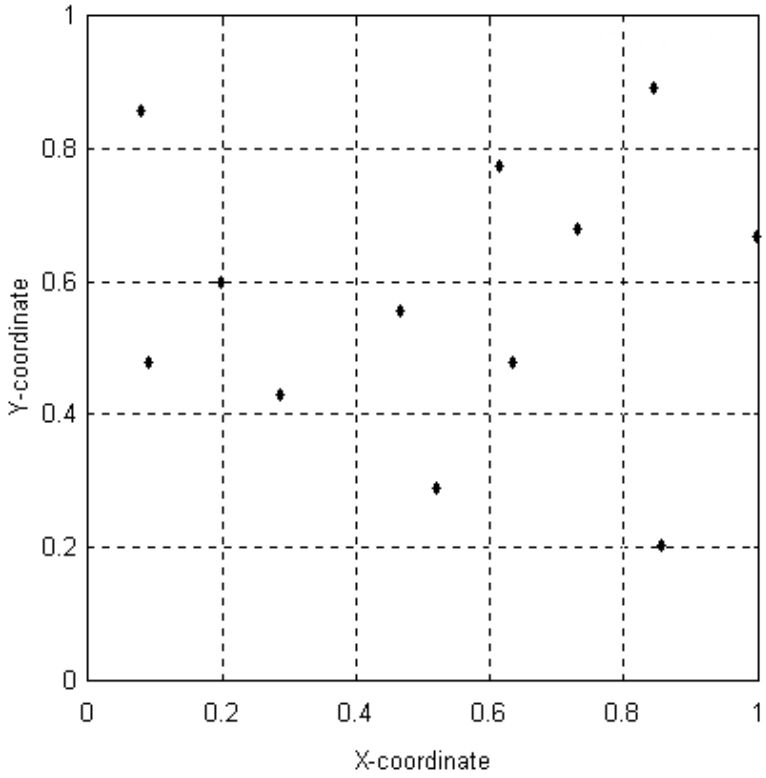


Figure 6.1. A traveling salesman problem with 12 cities.

Traveling Salesman Problem (TSP) aims at finding a route through all cities by visiting each exactly once and returning to the initial city such that the total distance traveled is minimized. Assume m artificial ants travel through n cities. The operation of ant systems for the traveling salesman problem (TSP) is given below (Dorigo et al. 1996, Dorigo & Gambardella 1997):

Step 1: Randomly select the initial city for each ant. The initial pheromone level between any two cities is set to be a small positive constant . Set the cycle counter to be 0.

Step 2: Calculate the transition probability from city r to city s for the kth ant as

$$P_k(r,s) = \begin{cases} \frac{[\tau(r,s)]\cdot[\eta(r,s)]^\beta}{\sum_{u\in J_k(r)}[\tau(r,u)]\cdot[\eta(r,u)]^\beta} & , \quad if\ s \in J_k(r) \\ 0 & , \quad otherwise \end{cases} \tag{6.1}$$

where r is the current city, s is the next city, $\tau(r,s)$ is the pheromone level between city r and city s, $\eta(r,s) = \frac{1}{\delta(r,s)}$ the inverse of the distance $\delta(r,s)$ between city r and city s, $J_k(r)$ is the set of cities that remain to be visited by the kth ant positioned on city r, and β is a parameter which determines the relative importance of pheromone level versus distance. Select the next visited city s for the kth ant with the probability $P_k(r,s)$. Repeat step 2 for each ant until the ants have toured all cities.

Step 3: Update the pheromone level between cities as

$$\tau(r,s) \longleftarrow (1-\alpha)\cdot\tau(r,s) + \sum_{k=1}^{m}\Delta\tau_k(r,s) \tag{6.2}$$

$$\Delta\tau_k(r,s) = \begin{cases} \frac{1}{L_k} & , \quad if\ (r,s) \in route\ done\ by\ ant\ k \\ 0 & , \quad otherwise \end{cases} \tag{6.3}$$

$\Delta\tau_k(r,s)$ is the pheromone level laid down between cities r and s by the kth ant, L_k is the length of the route visited by the kth ant, m is the number of ants and $0 < \alpha < 1$ is a pheromone decay parameter.

Step 4: Increment cycle counter. Move the ants to the originally selected cities and continue Steps 2 to 4 until the behavior stagnates or the maximum number of cycles has reached, where a stagnation is indicated when all ants take the same route.

From *Eq.* 6.1 it is clear Ant System (AS) needs a high level of computation to find the next visited city for each ant.

An implementation of the ant system by applying program to the test problem in Figure 6.1 are given in Figure 6.2 and 6.3. Figure 6.2 reports a found shortest route of length 3.308, which is the truly shortest route validated by exhaustive search. Figure 6.3 gives a snapshot of the pheromone intensities after 20 episodes. A higher intensity is represented by a wider edge. Notice that intensity alone cannot be used as a criteria for judging whether a link is a constitute part of the

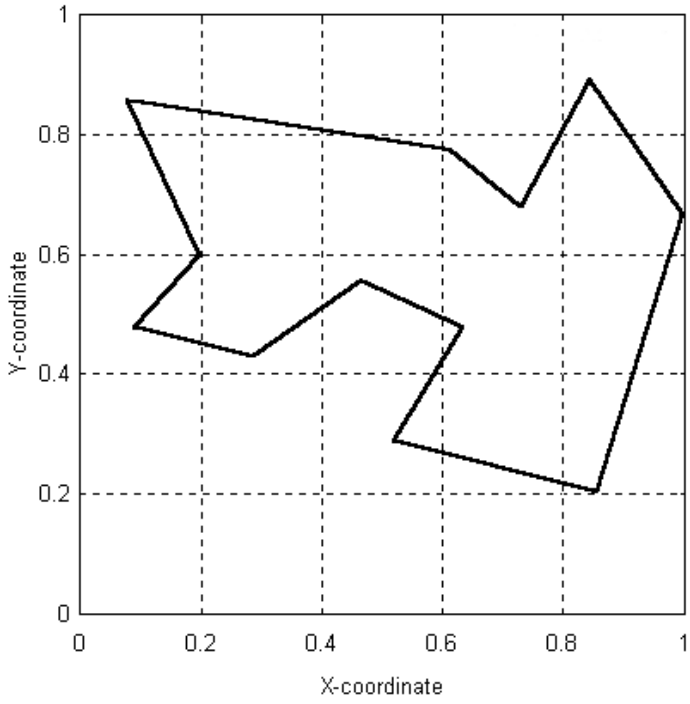


Figure 6.2. The shortest route found by the ant system.

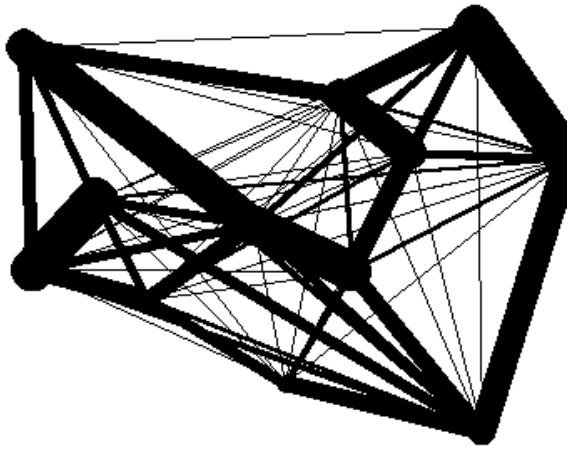


Figure 6.3. The snapshot of pheromone intensities after 20 episodes.

shortest route or not, since the shortest route relies on the cooperation of other links.

In order to improve the search efficiency, the Ant Colony System (ACS) was proposed (Dorigo & Gambardella 1997). ACS is based on AS but updates the pheromone level before moving to the next city (local updating rule) and updating the pheromone level for the shortest route only after completing the route for each ant (global updating rule). The algorithm of the Ant Colony System can be described as follows:

Step 1: Randomly select the initial city for each ant. The initial pheromone level between any two cities is set to be a small positive constant τ_0. Set the cycle counter to be 0.

Step 2: Calculate the next city s to be visited for each ant according to

$$s = \begin{cases} arg\, max_{u \in J_k(r)}[\tau(r,u)] \cdot [\eta(r,u)]^{\beta} & , \quad if\ q \leq q_0 \quad (exploitation) \\ S & , \quad otherwise \quad (biased\ exploration) \end{cases} \tag{6.4}$$

where q is a random number between 0 and 1, q_0 is a constant between 0 and 1, S is random variable selected using the probability distribution given in $Eq.$ 6.1 and β is a parameter which determines the relative importance of pheromone level versus distance.

Step 3: Update the pheromone level between cities as

$$\tau(r,s) \longleftarrow (1-\rho) \cdot \tau(r,s) + \rho \cdot \Delta\tau(r,s) \tag{6.5}$$

where $\Delta\tau(r,s) = \tau_0 = (n * L_{nn})^{-1}$ and L_{nn} is an approximate distance of the route of all cities using Nearest Neighbour Heuristic, n is the number of cities and $0 < \rho < 1$ is a pheromone decay parameter. Repeat Steps 2 and 3 for each ant until all cities are visited.

Step 4: Increment cycle counter. Update the pheromone level of the shortest route according to

$$\tau(r,s) \longleftarrow (1-\alpha) \cdot \tau(r,s) + \alpha \cdot \Delta\tau(r,s) \tag{6.6}$$

$$\Delta\tau(r,s) = \begin{cases} (L_{gb})^{-1} & , \quad if\ (r,s) \in global\ best\ route \\ 0 & , \quad otherwise \end{cases} \tag{6.7}$$

where L_{gb} is the length of the shortest route and α is a pheromone decay parameter. Move the ants to the originally selected cities and continue Steps 2 to 4 until the stagnates behavior or a maximum number of cycles has reached, as before.

6.2 Ant Colony System with Communication Strategies

A parallel computer consists of a large number of processing elements which can be dedicated to solving a single problem at a time. Pipeline processing and data parallelism are two popular parallel processing methods. The function of the pipeline processing is to separate the problem into a cascade of tasks where each task is executed by an individual processor, while data parallelism involves distributing the data to be processed amongst all processors which then executes the same procedure on each subset of the data. Data parallelism has been applied to genetic algorithm by dividing the population into several groups and running the same algorithm over each group using different processor (Cohoon et al. 1987). The resulting parallel genetic algorithm has been successfully applied to noise reduction of vector quantization based communication (Pan et al. 1996*a*).

6.2.1 Description

The Ant Colony System has been shown to be the improved version of Ant System by adding the local updating pheromone level immediately after moving each city for each ant and global updating pheromone level for the best route. we apply the idea of data parallelism to Ant Colony System (*ACS*) and a *Parallel Ant*

Colony System (PACS) (Chu, Roddick & Pan 2004*a*) is proposed. The Parallel Ant Colony System ($PACS$) is described as follows:

Step 1: Initialization – Generate N_j artificial ants for the jth group, $j = 0, 1 \ldots G-1$. N_j and G are the number of artificial ants for the jth group and the number of groups, respectively. Randomly select an initial city for each ant. The initial pheromone level between any two cities is set to be a small positive constant τ_0. Set the cycle counter to be 0.

Step 2: Movement – Calculate the next visited city s for the ith ant in the jth group according to

$$s = arg\, max_{u \in J_{i,j}(r)}[\tau_j(r,u)] \cdot [\eta(r,u)]^{\beta}, \quad if\ q \leq q_0 \quad (exploitation)$$

$$visit\ city\ s\ with\ P_{i,j}(r,s), \quad if\ q > q_0 \quad (biased\ exploration)$$

$$P_{i,j}(r,s) = \begin{cases} \frac{[\tau_j(r,s)] \cdot [\eta(r,s)]^{\beta}}{\sum_{u \in J_k(r)} [\tau_j(r,u)] \cdot [\eta(r,u)]^{\beta}} & , \quad if\ s \in J_{i,j}(r) \\ 0 & , \quad otherwise \end{cases}$$

where $P_{i,j}(r,s)$ is the transition probability from city r to city s for the ith ant in the jth group. $\tau_j(r,s)$ is the pheromone level between city r to city s in the jth group. $\eta(r,s) = \frac{1}{\delta(r,s)}$ the inverse of the distance $\delta(r,s)$ between city r and city s. $J_{i,j}(r)$ is the set of cities that remain to be visited by the ith ant in the jth group and β is a parameter which determines the relative importance of pheromone level versus distance. q is a random number between 0 and 1 and q_0 is a constant between 0 and 1.

Step 3: Local Pheromone Level Updating Rule – Update the pheromone level between cities for each group as

$$\tau_j(r,s) \longleftarrow (1-\rho) \cdot \tau_j(r,s) + \rho \cdot \Delta\tau(r,s)$$

$$\Delta\tau(r,s) = \tau_0 = (n * L_{nn})^{-1}$$

where $\tau_j(r,s)$ is the pheromone level between cities r and s for the ants in the jth group, L_{nn} is an approximate distance of the route between all

cities using the *Nearest Neighbour Heuristic*, n is the number of cities and $0 < \rho < 1$ is a pheromone decay parameter. Continue Steps 2 and 3 until each ant in each group completes the route.

Step 4: Evaluation – Calculate the total length of the route for each ant in each group.

Step 5: Global Pheromone Level Updating Rule – Update the pheromone level between cities for each group as

$$\tau_j(r,s) \longleftarrow (1-\alpha)\cdot\tau_j(r,s) + \alpha\cdot\Delta\tau_j(r,s)$$

$$\Delta\tau_j(r,s) = \begin{cases} (L_j)^{-1} & , \quad if\ (r,s) \in best\ route\ of\ jth\ group \\ 0 & , \quad otherwise \end{cases}$$

where L_j is the shortest length for the ants in the jth group and α is a pheromone decay parameter.

Step 6: Updating From Communication – Seven communication strategies are proposed as follows:

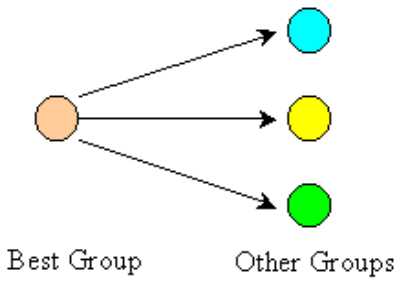


Figure 6.4: Update the pheromone level according to the best route of all groups

- **Strategy 1:** As shown in Figure 6.4, update the pheromone level between cities for each group for every R_1 cycles as

$$\tau_j(r,s) \longleftarrow \tau_j(r,s) + \lambda\cdot\Delta\tau_{best}(r,s)$$

$$\Delta\tau_{best}(r,s) = \begin{cases} (L_{gb})^{-1} & , \quad if\ (r,s) \in best\ route\ of\ all\ groups \\ 0 & , \quad otherwise \end{cases}$$

where λ is a pheromone decay parameter and L_{gb} is the length of the best route of all groups, i.e., $L_{gb} \leq L_j$, $j = 0, 1 \ldots G-1$.

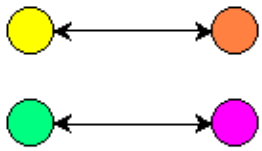


Figure 6.5. Update the pheromone level between each pair of groups

- **Strategy 2:** As shown in Figure 6.5, update the pheromone level between cities for each group for every R_2 cycles as

$$\tau_j(r,s) \longleftarrow \tau_j(r,s) + \lambda \cdot \Delta\tau_{ng}(r,s)$$

$$\Delta\tau_{ng}(r,s) = \begin{cases} (L_{ng})^{-1} & , \quad if\ (r,s) \in best\ route\ of\ neighbour\ group \\ 0 & , \quad otherwise \end{cases}$$

where *neighbour* is defined as being the group whose binary representation of the group number j differs by the least significant bit. λ is a pheromone decay parameter and L_{ng} is the length of the shortest route in the neighbour group.

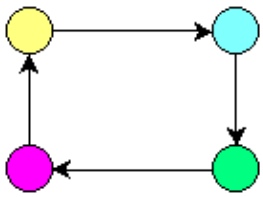


Figure 6.6: Update the pheromone level according to the ring structure

- **Strategy 3:** As shown in Figure 6.6, update the pheromone between

cities for each group for every R_3 cycles as

$$\tau_j(r,s) \longleftarrow \tau_j(r,s) + \lambda \cdot \Delta\tau_{ng}(r,s)$$

$$\Delta\tau_{ng}(r,s) = \begin{cases} (L_{ng})^{-1} & , \quad if\ (r,s) \in best\ route\ of\ neighbour\ group \\ 0 & , \quad otherwise \end{cases}$$

where *neighbour* is defined as being the group arranged as the ring structure. λ is a pheromone decay parameter and L_{ng} is the length of the shortest route in the neighbour group.

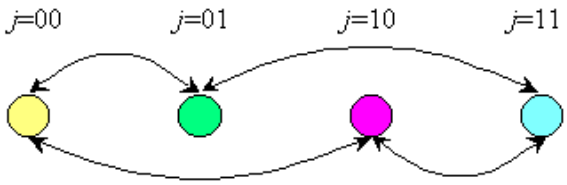


Figure 6.7: Update the Pheromone level to the neighbours according to the group number j differs by one bit

- **Strategy 4:** As shown in Figure 6.7, update the pheromone between cities for each group for every R_4 cycles as

$$\tau_j(r,s) \longleftarrow \tau_j(r,s) + \lambda \cdot \Delta\tau_{ng}(r,s)$$

$$\Delta\tau_{ng}(r,s) = \begin{cases} (L_{ng})^{-1} & , \quad if\ (r,s) \in best\ route\ of\ neighbour\ group \\ 0 & , \quad otherwise \end{cases}$$

where *neighbour* is defined as being those groups where the binary representation of the group number j differs by one bit. λ is a pheromone decay parameter and L_{ng} is the length of the shortest route in the neighbour group.

- **Strategy 5:** Update the pheromone between cities for each group using both Strategy 1 and Strategy 2.
- **Strategy 6:** Update the pheromone between cities for each group using both Strategy 1 and Strategy 3.

- **Strategy 7:** Update the pheromone between cities for each group using both Strategy 1 and Strategy 4.

Step 7: Termination – Increment the cycle counter. Move the ants to the originally selected cities and continue Steps 2 to 6 until the stagnation or a present maximum number of cycles has reached, where a stagnation indicated by all ants taking the same route.

6.2.2 Experimental Results

To evaluate the effectiveness of $PACS$, we have performed an extensive performance study. In this section, we report our experimental results on comparing $PACS$ with Ant System (AS) and Ant Colony System (ACS). It is shown that $PACS$ and various combinations outperform both Ant System (AS) and Ant Colony System (ACS).

We used three generally available and typical data sets, EIL101, ST70 and TSP225 as the test material [1] to test the performance of the Ant System (AS), Ant Colony System (ACS) and Parallel Ant Colony System ($PACS$) for the traveling salesman problem.

To ensure a fair comparison among AS, ACS and $PACS$, *the number of groups* $\times$ *the number of ants per group* was kept constant – the number of ants for AS and ACS were set to be 80, one swarm with 80 ants, as reported by 1×80. For $PACS$, the number of ants was also set to be 80 that was divided into 4 groups with 20 ants in each group (i.e. 4×20) and 8 groups with 10 ants in each group (i.e. 8×10), respectively. The parameters were set to the following values: $\beta = 2$, $q_0 = 0.9$, $\alpha = \rho = \lambda = 0.1$ (Dorigo & Gambardella 1997). The number of iterations for both EIL101 and ST70 were set to be 1000 and TSP225 was set to be 2000 as the cities of TSP225 are more than EIL101 and ST70 data

[1] available from *http://www.iwr.uniheidelberg.de/groups/comopt/software/*

sets. The number of cycles (i.e. R_1, R_2, R_3 and R_4) between updates of the pheromone level from communication for strategies 1 to 7 in $PACS$ were set to be 30. In order to test the performance of the different approaches to the traveling salesman problem, variously proposed communication strategies for updating the pheromone level between groups in $PACS$ were combined. Where appropriate, these seven communication strategies are applied to the $PACS$ and compared to AS and ACS.

EIL101, ST70 and TSP225 are data sets with 101, 70 and 225 cities, respectively. Experimental results were carried out to the average shortest length for 10 seeds. The performance of $PACS$ (i.e. ACS with communication strategy) is better by in comparison with AS and ACS can be illustrated by Figure 6.8, 6.9 and Figure 6.10. As can be seen from the Table 6.1, Table 6.2 and Table 6.3, $PACS$ outperforms both AS and ACS on effectiveness.

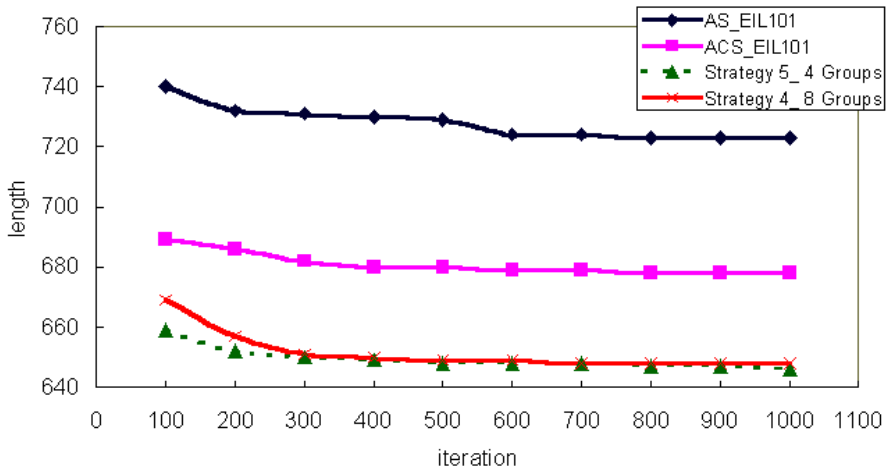


Figure 6.8: Performance comparison among AS, ACS and two arbitrarily chosen strategies for EIL101 data set.

The EIL101 data set was used for the first experiment. As shown in Table 6.1, the average improvement on EIL101 for proposed strategy 5 for 4 groups with 20 ants in each group by comparing with AS and ACS were much better up to be

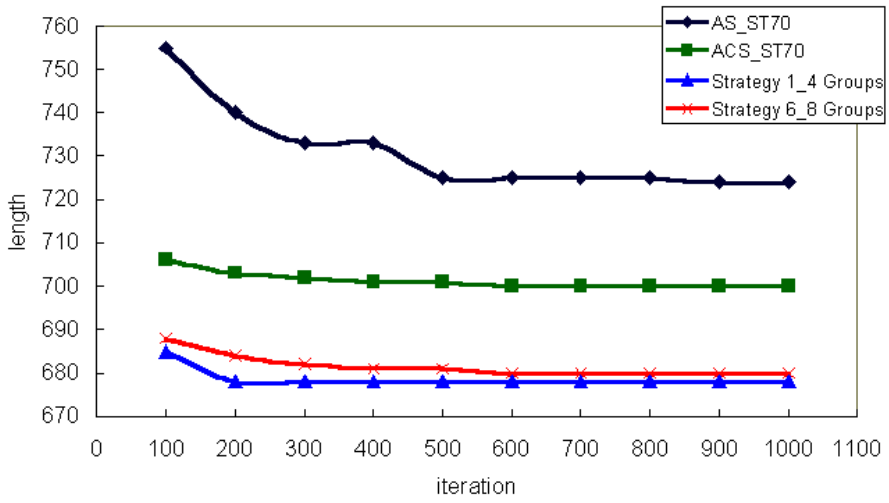


Figure 6.9: Performance comparison among AS, ACS and two arbitrarily chosen strategies for ST70 data set

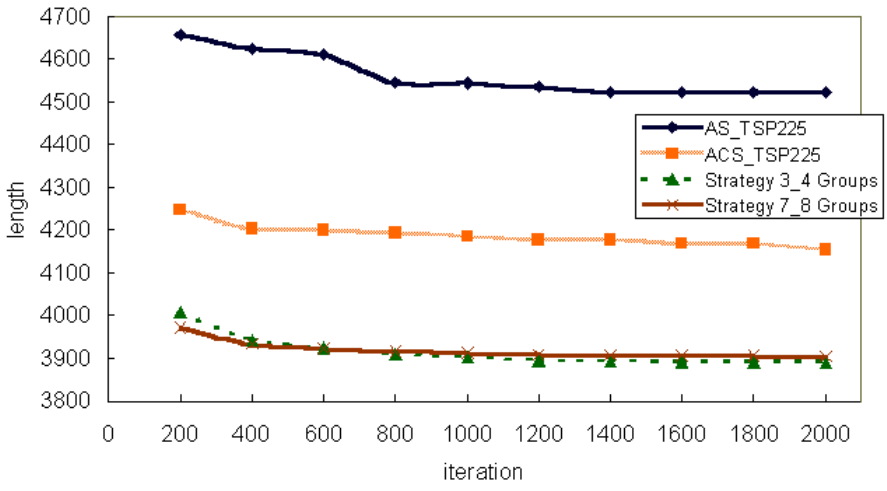


Figure 6.10: Performance comparison among AS, ACS and two arbitrarily chosen strategies for TSP225 data set

10.57% and 4.70%, respectively. In comparison with AS and ACS, the average improvement on EIL101 for proposed strategy 3 for 8 groups with 10 ants in each

group were 10.41% and 4.52%, respectively.

The ST70 data set was used for the second experiment. As we can see from Table 6.2, the average performance of proposed strategy 3 for 4 groups with 20 ants in each group by compared with AS and ACS were 6.20% and 3.06%, respectively and that of proposed strategy 6 for 8 groups with 10 ants in each group were 6.06% and 2.92%, respectively.

Finally, in the case of TSP225 data set, the experimental results shown in Table 6.3, compared with AS and ACS, shows that the average performance of proposed strategy 3 for 4 groups with 20 ants in each group were 13.97% and 6.35%, respectively and that of proposed strategy 5 for 8 groups with 10 ants in each group were 14.06% and 6.44%, respectively.

The main contribution of this section is to propose the parallel formulation for the Ant Colony System (ACS). Seven communication strategies between groups which can be used to update the pheromone levels are presented. For our preliminary experiments, the proposed Parallel Ant Colony System ($PACS$) outperforms both ACS and AS based on three available traveling salesman data sets. In general, our presented systems based on data set with large data can get much better performance such that the average improvement of TSP225 is better than that of ST70. The proposed $PACS$ may be applied to solve the quadratic assignment problem (Maniezzo & Colorni 1999), data mining (Parpinelli et al. 2002), space-planning (Bland 1999), data clustering and the combinatorial optimization problems.

Table 6.1: The performance of ACS with communication strategies (strategy 1 $\sim$ 7) obtained in comparison with AS and ACS for EIL101 data set on TSP problem

Seed	AS	ACS	$Strategy1$		$Strategy2$		$Strategy3$		$Strategy4$		$Strategy5$		$Strategy6$		$Strategy7$	
	1,80	1,80	4,20	8,10	4,20	8,10	4,20	8,10	4,20	8,10	4,20	8,10	4,20	8,10	4,20	8,10
1	730	683	657	655	648	653	649	645	654	644	646	651	646	653	647	646
2	730	680	657	655	650	647	643	648	647	649	641	650	660	643	648	643
3	731	681	644	646	655	655	641	646	653	646	641	648	646	645	648	642
4	720	678	645	648	651	654	651	647	647	647	643	646	642	647	650	652
5	727	676	641	643	648	663	648	656	651	651	644	650	647	651	647	647
6	727	673	656	655	648	644	645	655	648	649	647	653	651	653	644	655
7	698	675	642	644	649	658	646	648	651	650	650	646	647	648	645	645
8	726	679	646	651	653	662	658	645	647	645	653	650	647	653	652	655
9	721	672	645	646	651	656	652	642	649	650	652	647	649	651	646	650
10	718	685	643	651	654	647	645	645	652	651	646	647	646	652	650	648
Average	723	678	648	649	651	654	648	648	650	648	646	649	648	650	648	648

Table 6.2: The performance of ACS with communication strategies (strategy 1 $\sim$ 7) obtained in comparison with AS and ACS for ST70 data set on TSP problem

Seed	AS	ACS	$Strategy1$		$Strategy2$		$Strategy3$		$Strategy4$		$Strategy5$		$Strategy6$		$Strategy7$	
	1,80	1,80	4,20	8,10	4,20	8,10	4,20	8,10	4,20	8,10	4,20	8,10	4,20	8,10	4,20	8,10
1	734	701	679	680	683	684	678	683	681	683	684	679	680	677	682	681
2	721	700	681	681	677	686	688	681	677	681	679	681	677	681	677	681
3	722	700	681	682	678	683	678	681	678	681	677	681	681	682	678	677
4	717	701	677	688	682	687	678	682	679	694	685	683	686	681	681	689
5	721	703	678	684	678	686	678	681	678	678	678	679	682	683	677	682
6	713	702	691	690	694	683	678	682	678	678	692	681	689	684	689	686
7	714	700	682	683	678	683	677	683	681	677	681	682	678	677	677	677
8	730	701	677	682	677	679	677	682	681	683	681	677	678	678	678	683
9	730	696	677	681	679	685	678	678	677	682	683	679	683	677	678	681
10	736	699	678	693	682	688	678	681	681	682	680	691	682	678	677	690
Average	724	700	680	684	681	684	679	681	679	682	682	681	682	680	680	683

Table 6.3: The performance of ACS with communication strategies (strategy 1 $\sim$ 7) obtained in comparison with AS and ACS for TSP225 data set on TSP problem

Seed	AS	ACS	$Strategy1$		$Strategy2$		$Strategy3$		$Strategy4$		$Strategy5$		$Strategy6$		$Strategy7$	
	1,80	1,80	4,20	8,10	4,20	8,10	4,20	8,10	4,20	8,10	4,20	8,10	4,20	8,10	4,20	8,10
1	4587	4145	3907	3933	3913	3914	3879	3885	3903	3949	3882	3905	3866	3905	3884	3943
2	4492	4215	3903	3883	3879	3879	3883	3877	3955	3942	3916	3871	3902	3892	3891	3881
3	4454	4149	3888	3926	3900	3900	3889	3896	3953	3916	3906	3888	3878	3894	3919	3902
4	4609	4160	3892	3886	3908	3952	3889	3885	3895	3890	3871	3885	3866	3879	3919	3899
5	4538	4163	3881	3869	3888	3898	3879	3885	3879	3880	3882	3910	3878	3884	3886	3922
6	4483	4146	3942	3915	3916	3978	3892	3901	3961	3895	3883	3877	3901	3866	3882	3882
7	4555	4149	3904	3911	3876	3939	3881	3891	3881	3887	3881	3876	3892	3885	3882	3912
8	4491	4148	3950	3900	3912	3925	3950	3889	3890	3902	3957	3891	3936	3950	3952	3903
9	4500	4108	3903	3916	3903	3904	3889	3887	3896	3886	3882	3891	3904	3875	3881	3898
10	4521	4161	3877	3915	3875	3911	3877	3896	3876	3873	3884	3876	3919	3917	3895	3909
Average	4523	4154	3905	3905	3897	3920	3891	3889	3909	3902	3894	3887	3894	3895	3899	3905

6.3 Adaptive Ant Colony System for Data Clustering

Processes that simulate natural phenomena have successfully been applied to a number of problems for which no simple mathematical solution is known or is practicable. Such meta-heuristic algorithms include genetic algorithms, particle swarm optimization and ant colony systems and have received increasing attention in recent years.

In this section, an advanced version of the ACO algorithm, termed the Constrained Ant Colony Optimization ($CACO$) algorithm, is proposed here for data clustering by adding constrains on the calculation of pheromone strength. The proposed $CACO$ algorithm has the following properties:

- It applies the quadratic metric combined with the Sum of K Nearest Neighbor Distances ($SKNND$) metric to be instead of the Euclidean distance measure.
- It adopts a constrained form of pheromone updating. The pheromone is only updated based on some statistical distance threshold.
- It utilises a reducing search range.

6.3.1 Ant Colony Optimization with Different Favor ($ACODF$)

Ant Colony Optimization with Different Favor ($ACODF$) algorithm (Tsai, Wu & Tsai 2002) modified the Ant Colony Optimization (ACO) (Dorigo & Gambardella 1997) for data clustering by adding the concept of simulated annealing (Kirkpatrick et al. 1983) and the strategy of tournament selection (Brindle 1981). It is useful in partitioning the datasets for the clear boundaries among clusters, however, it is not suitable to partition the datasets for the clusters with arbitrary shapes, clusters with outliers and bridges between clusters.

Ant Colony Optimization with Different Favor ($ACODF$) applies ACO for use in data clustering. The difference between the $ACODF$ and ACO is that each ant in $ACODF$ only visits a fraction of the total clustering objects and the number of visited objects decreases with each cycle. $ACODF$ also incorporates the strategies of simulated annealing and tournament selection and results in an algorithm which is effective for clusters with clearly defined boundaries. However, $ACODF$ does not handle clusters with arbitrary shapes, clusters with outliers and bridges between clusters well. In order to improve the effectiveness of the clustering based on the technique of Ant Colony Optimization, our proposed $CACO$ algorithm may solve these problems of clusters with arbitrary shapes, clusters with outliers and bridges between clusters.

6.3.2 The Constrained Ant Colony Optimization *(CACO)*

An advanced version of Ant Colony Optimization algorithm termed Constrained Ant Colony Optimization ($CACO$) algorithm was proposed for data clustering by adding constrains for computing the pheromone strength. The $CACO$ algorithm extends the Ant Colony Optimization algorithm by accommodating a quadratic metric, Sum of K Nearest Neighbor Distances ($SKNND$), constrained addition of pheromone and a shrinking range strategy to improve the data clustering (Pan, Chu, Roddick & Su 2004). In order to improve the effectiveness of the clustering the following four strategies are applied:

Strategy 1: While the Euclidean distance measure is used in conventional clustering techniques such as in the $ACODF$ clustering algorithm, it is not suitable for clustering non-spherical clusters, (for example, a cluster with a slender shape). In this work we therefore opt for a quadratic metric (Pan et al. 1996*b*) as the distance measure. Given an object at position O and objects X_i, $i = 1, 2, \ldots, T$, (T is the total number of objects), the quadratic metric between the current object O and the object X_m can be expressed

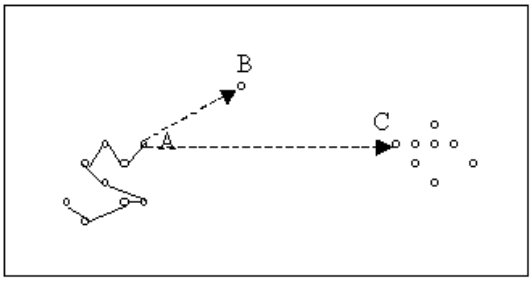


Figure 6.11. Ant tends moving toward the object with dense cluster

as

$$D_q(O, X_m) = (O - X_m)^t W^{-1} (O - X_m) \tag{6.8}$$

where $(O - X_m)$ is error column vector and W is the covariance matrix given as

$$W = \frac{1}{T} \sum_{i=1}^{T} (X_i - \bar{X})(X_i - \bar{X})^t \tag{6.9}$$

here $\bar{X}$ is the mean of X_i, $i = 1, 2, \cdots, T$ defined as

$$\bar{X} = \frac{1}{T} \sum_{i=1}^{T} X_i \tag{6.10}$$

W^{-1} is the inverse of covariance matrix W.

Strategy 2: We use the *Sum of K Nearest Neighbor Distances* ($SKNND$) metric in order to distinguish dense clusters more easily. The example shown in Figure 6.11 shows an ant located at A which will tend to move toward C within a dense cluster rather than object B located in the sparser region. By adopting $SKNND$, as the process iterates, the probability for an ant to move towards the denser clusters increases. This strategy can avoid clustering errors due to bridges between clusters.

Strategy 3: As shown in Figure 6.11, as a result of strategy 2, ants will tend to move towards denser clusters. However, the pheromone update is inversely proportional to the distance between the visited objects as shown in Equation 6.5 and Equation 6.6 and the practical distance between objects

A and C could be farther than that between objects A and B reducing the pheromone level and causing a clustering error. In order to compensate for this, a statistical threshold for the k^{th} ant is adopted as below.

$$L_{ts}^{k} = AvgL_{path}^{k} + StDevL_{path}^{k} \tag{6.11}$$

where $AvgL_{path}^{k}$ and $StDevL_{path}^{k}$ are the average of the distance and the standard deviation for the route of the visited objects by the kth ant expressed as

$$AvgL_{path}^{k} = \frac{\sum L_{ij}^{k}}{E}, \quad if\ (X_i, X_j)\ path\ visited\ by\ the\ kth\ ant \tag{6.12}$$

$$StDevL_{path}^{k} = \sqrt{\frac{\sum (L_{ij}^{k} - AvgL_{path}^{k})^2}{E}},$$

$$if\ (X_i, X_j)\ path\ visited\ by\ the\ kth\ ant \tag{6.13}$$

where E is the number of paths visited by the kth ant. We may roughly consider object X_i and object X_j are located in different clusters if $L_{ij}^{k} > L_{ts}^{k}$. The distance between object X_i and object X_j cannot be added into the length of the path and the pheromone cannot be updated between the objects.

Strategy 4: Equation 6.1 is the conventional search formula between objects r and s for ant colony optimization. However this formula is not suitable for robust clustering as object s represents all un-visited objects resulting in excessive computation and a tendency for ants to jump between dense clusters as shown in Figure 6.12. In order to improve clustering speed and eliminate this jumping phenomenon, Equation 6.1 is modified to be

$$P_k(r,s) = \begin{cases} \frac{[\tau(r,s)]\cdot[D_q(r,s)]^{-\beta}\cdot[SKNND(s)]^{-\gamma}}{\sum_{u\in J_k^{N_2}(r)}[\tau(r,u)]\cdot[D_q(r,u)]^{-\beta}\cdot[SKNND(u)]^{-\gamma}} & , \quad if\ s \in J_k^{N_2}(r) \\ 0 & , \quad otherwise \end{cases} \tag{6.14}$$

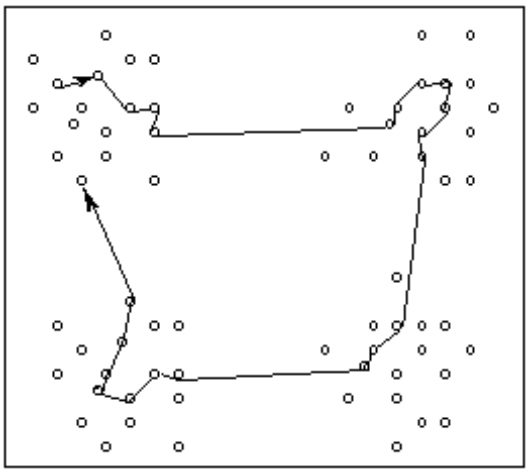


Figure 6.12. Conventional search route using Equation 6.1

where $J_k^{N_2}(r)$ is to shrink the search range to N_2 nearest un-visited objects. N_2 is set to be 1/10 objects. $D_q(r,s)$ is the quadratic distance between object r and object s. $SKNND(s)$ is the Sum of the distance between the object s and the N_2 nearest objects. β and γ are two parameters which determine the relative importance of pheromone level versus the quadratic distance and the Sum of N_2 Nearest Neighbor Distance, respectively. β is set to 2 and γ would be robust to set between 5 and 15. As shown in Figure 6.13, the jumping phenomenon is deleted after using the shrinking search formula.

The Constrained Ant Colony Optimization algorithm for data clustering can be expressed as follows:

Step 1: Initialization

Randomly select the initial object for each ant. The initial pheromone τ_{ij} between any two objects X_i and X_j is set to be a small positive constant τ_0.

Step 2: Movement

Let each ant moves to N_1 objects only using Equation 6.14. Here N_1 is set to be 1/20 data objects.

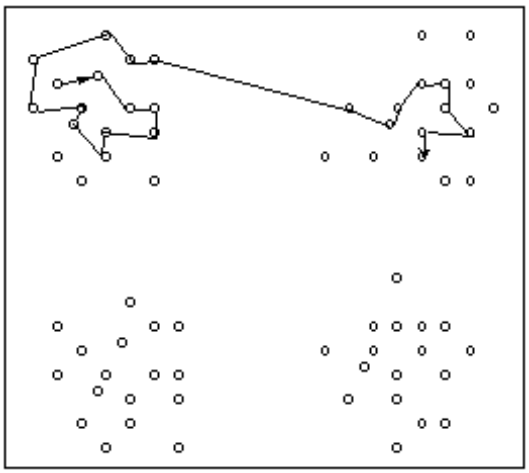


Figure 6.13. Shrinking search route using Equation 6.14

Step 3: Pheromone Update

Update the pheromone level between objects as

$$\tau_{ij}(t+1) = (1-\alpha)\tau_{ij}(t) + \Delta\tau_{ij}(t+1) \tag{6.15}$$

$$\Delta\tau_{ij}(t+1) = \sum_{k=1}^{T} \Delta\tau_{ij}^{k}(t+1) \tag{6.16}$$

$$\Delta\tau_{ij}^{k}(t+1) = \begin{cases} \frac{Q}{L_k} & , \quad if\ ((i,j) \in route\ done\ by\ ant\ k\ and\ L_{ij}^{k} < L_{ts}^{k} \\ 0 & , \quad otherwise \end{cases} \tag{6.17}$$

where τ_{ij} is the pheromone level between object X_i and object X_j, T is the total number of clustering objects, α is a pheromone decay parameter and Q is a constant and is set to 1. L_k is the length of the route after deleting the distance between object X_i and object X_j in which $L_{ij}^{k} > L_{ts}^{k}$ for the kth ant.

Step 4: Consolidation

Calculate the average pheromone level on the route for all objects as

$$Avg\tau = \frac{\sum_{i,j \in E} \tau_{ij}}{E} \tag{6.18}$$

where E is the number of paths visited by the kth ant. Disconnect the path between two objects if the pheromone level between these two objects

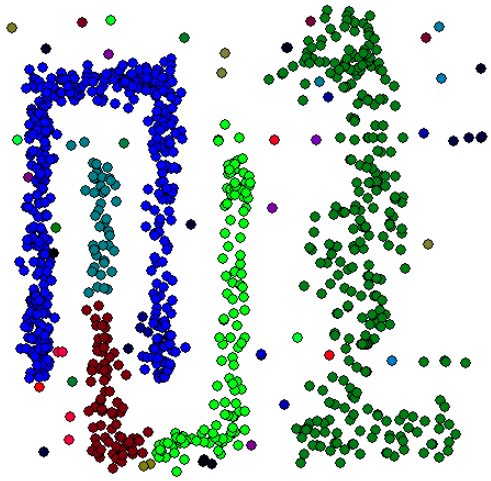


Figure 6.14. Clustering result of CACO ($N_1 = \frac{1}{55}$)

is smaller than $Avg\tau$. All the objects connected together are in the same cluster.

6.3.3 Experiments and Results

The experiments were carried out to test the performance of the data clustering for Ant Colony Optimization with Different Favor ($ACODF$), $DBSCAN$, $CURE$ and the proposed Constrained Ant Colony Optimization ($CACO$). Four data sets, Four-Cluster, Four-Bridge, Smile-Face and Shape-Outliers were used as the test material consisting of 892, 981, 877 and 999 objects, respectively.

In order to cluster a data set using $CACO$, N_1 and γ are two important parameters which will influence the clustering results. N_1 is the number of objects to be visited in each cycle for each ant. If N_1 is set too small, the ants cannot finish visiting all the objects belonged to the same cluster resulting in a division of slender shaped cluster into several sub-clusters as shown in Figure 6.14. Our experiments indicated that good experimental results were obtained by setting N_1 to 1/20 as shown in Figure 6.15.

γ also influences the clustering result for clusters with bridges or high numbers of outliers. As shown in Figure 6.16, the Four-Bridge data set will be partitioned

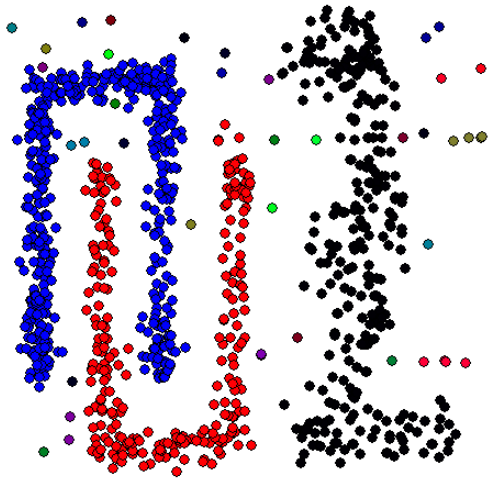


Figure 6.15. Clustering result of CACO ($N_1 = \frac{1}{20}$)

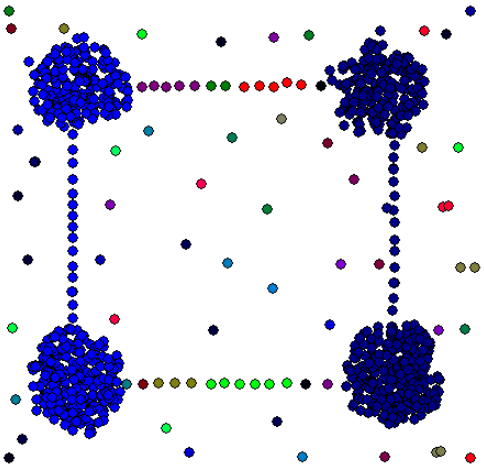


Figure 6.16. Clustering result of CACO ($\gamma = 1$)

into just two clusters if γ is set to 1. By setting γ to 5, the Four-Clusters data set can be correctly partitioned as shown in Figure 6.17. We found that γ set between 5 and 15 provided robust results.

DBSCAN is a well-known clustering algorithm that works well for clusters with arbitrary shapes. Following the recommendation of Ester *et al.* (1996), $MinPts$ was fixed to 4 and ϵ was changed during the experiments. *CURE* produces high-quality clusters in the existence of outliers, allowing complex shaped clusters and different size. We performed experiments with shrinking factor is

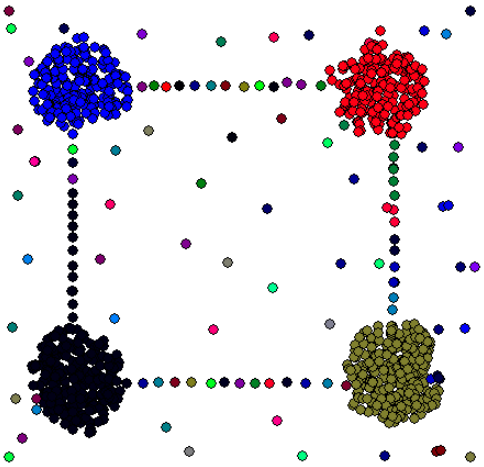


Figure 6.17. Clustering result of CACO ($\gamma = 5$)

0.3 and the number of representative points as 10, which are the default values recommended in Guha *et al.* (1998).

The first experiment tested the clustering performance of the Four-Cluster data set using the $ACODF$, $DBSCAN$, $CURE$ and $CACO$ algorithms. As is illustrated in Figure 6.18, the $ACODF$ algorithm was able to correctly cluster the Four-cluster data set. Figure 6.19 shows the clusters found by $CACO$ for Four-Cluster data set. For the results shown in Figure 6.20, $DBSCAN$ worked well when the $MinPts$ was fixed to 4 and $\epsilon = 8.357$. $CURE$ was able to find the right clusters, but some noises were present inside the clusters as shown in Figure 6.21.

The second experiment was to partition the Four-Bridge data sets. As we can see from Figures 6.22 to 6.25, the $ACODF$ algorithm puts the four spheres into the same clusters as the outlier points connecting these clusters while $DBSCAN$ cannot correctly separate these clusters. In Figure 6.23, $DBSCAN$ fails to perform well and puts the four spheres into two clusters because of the outlier points connecting these spheres. Although $CURE$ clusters the data set into four clusters, there is noise inside and around these clusters shown in Figure 6.24. The $CACO$ algorithm is able to separate this data set to four clusters as well as

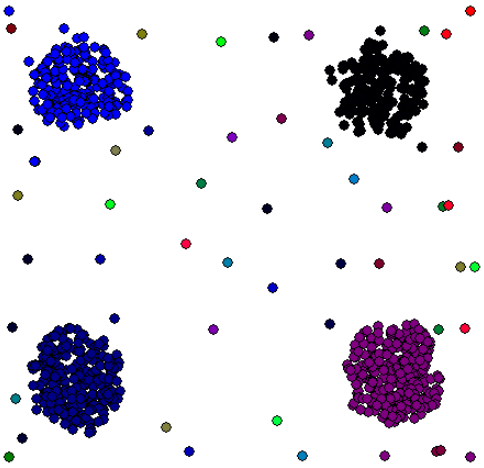


Figure 6.18: Clustering results of Four-Cluster by ACODF algorithm.

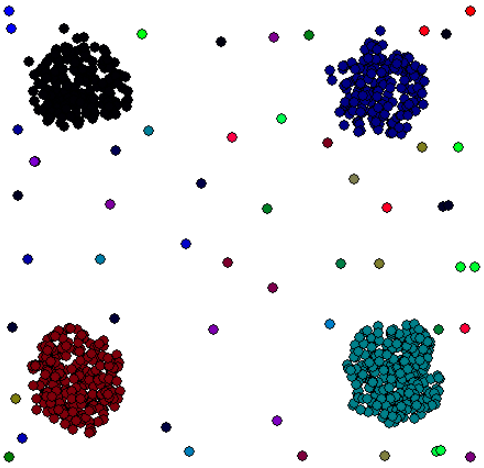


Figure 6.19. Clustering results of Four-Cluster by CACO algorithm.

identify the rest as outlier points connecting the four spheres.

The third experiment was to test the Smile-Face data set. Figure 6.26 shows the results obtained by $ACODF$ algorithm for Smile-Face data set, which partitions this data set as one cluster only. As shown in Figure 6.27, the results illustrate that $DBSCAN$ is able to find eyes, nose and mouth clusters but it fails to find the outline cluster as the outline cluster has a few fragments. $CURE$ cannot effectively find clusters shown in Figure 6.28 because the clusters in the data set are fragmented into a number of smaller clusters while the $CACO$ algorithm

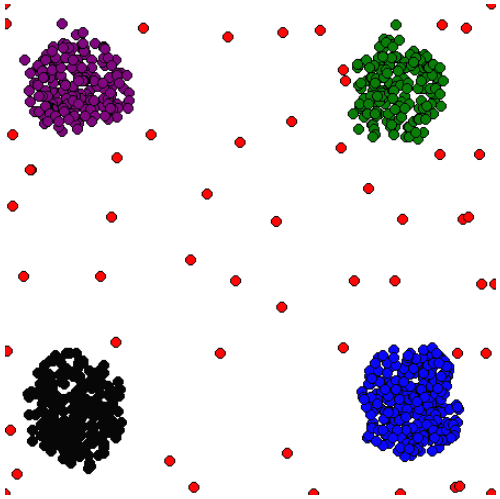


Figure 6.20: Clustering results of Four-Cluster by DBSCAN algorithm.

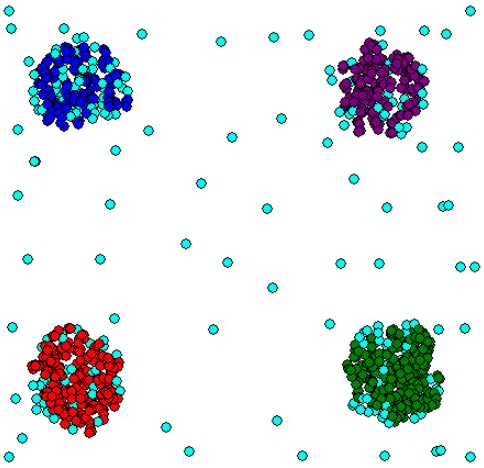


Figure 6.21. Clustering results of Four-Cluster by CURE algorithm.

can correctly partition the Smile-Face to five clusters shown in Figure 6.29.

The last experiment was to partition the Shape-Outliers data set. As in the previous experiment, the *ACODF* algorithm cannot correctly partition the Shape-Outliers data set shown in Figure 6.30. Figure 6.31 shows the clusters found by *DBSCAN*, but it also makes a mistake in that it has fragmented the clusters in the right-side 'L'-shaped cluster. Figure 6.32 shows that *CURE* fails to perform well on Shape-Outliers data set, with the clusters has fragmented into a number of smaller clusters. Looking at Figure 6.33, we can see that *CACO*

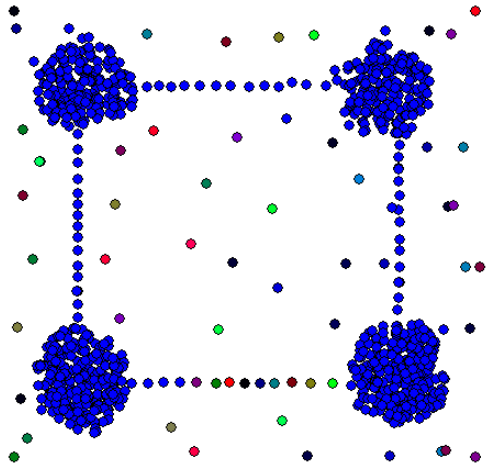


Figure 6.22. Clustering results of Four-Bridge by ACODF algorithm.

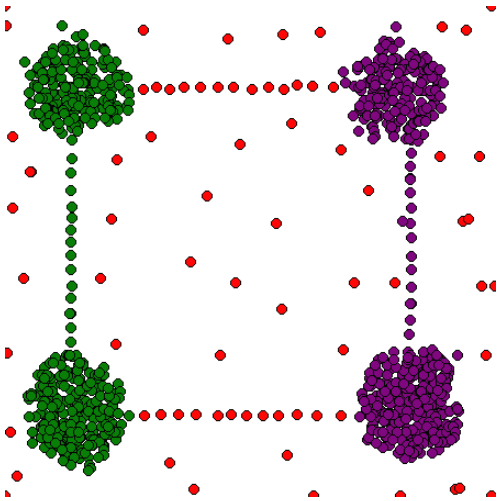


Figure 6.23: Clustering results of Four-Bridge by DBSCAN algorithm.

algorithm correctly identifies the clusters.

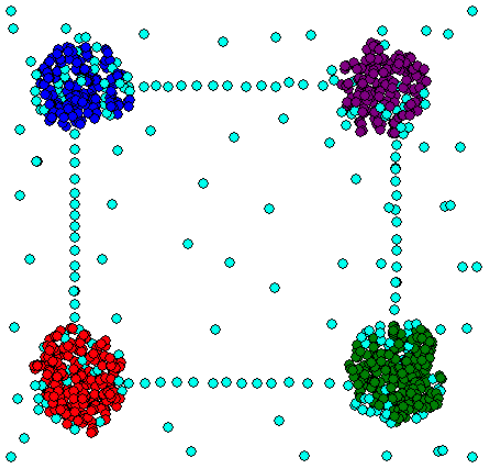


Figure 6.24. Clustering results of Four-Bridge by CURE algorithm.

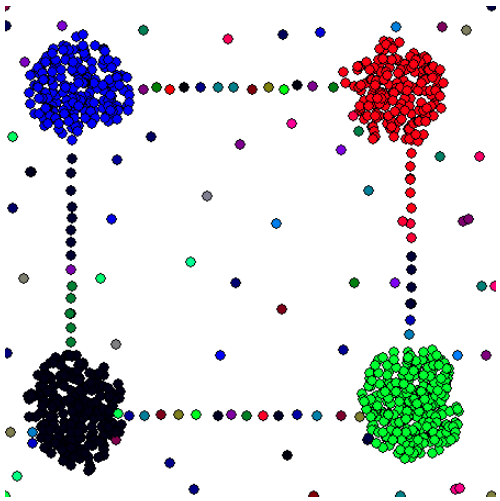


Figure 6.25. Clustering results of Four-Bridge by CACO algorithm.

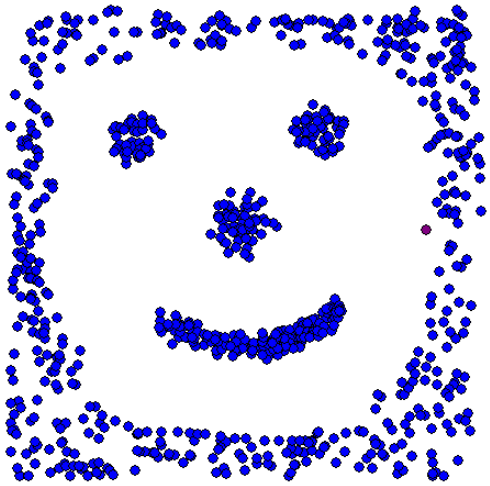


Figure 6.26. Clustering results of Smile-Face by ACODF algorithm.

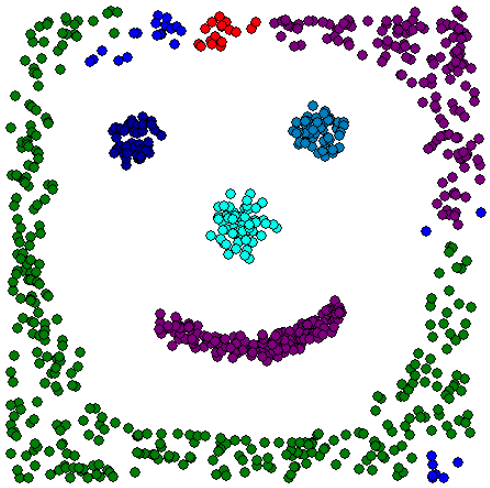


Figure 6.27. Clustering results of Smile-Face by DBSCAN algorithm.

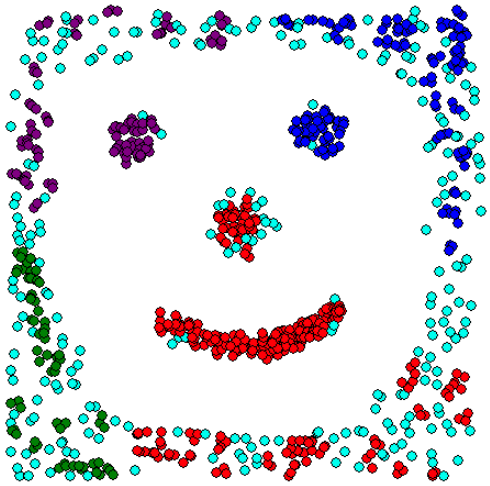


Figure 6.28. Clustering results of Smile-Face by CURE algorithm.

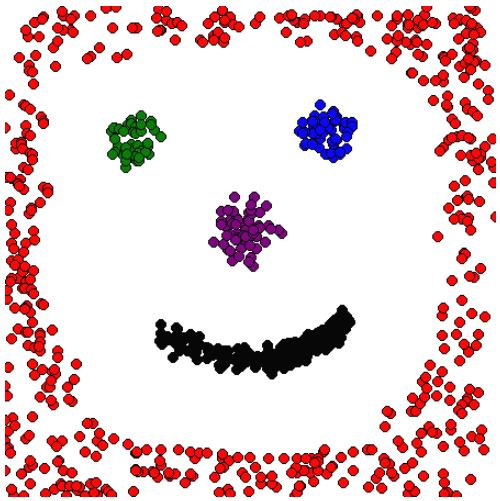


Figure 6.29. Clustering results of Smile-Face by CACO algorithm.

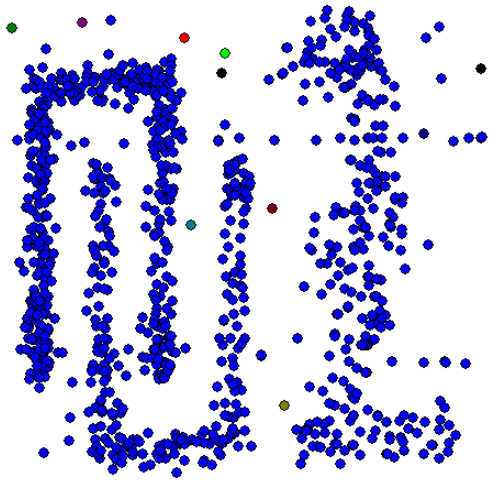


Figure 6.30: Clustering results of Shape-Outliers by ACODF algorithm.

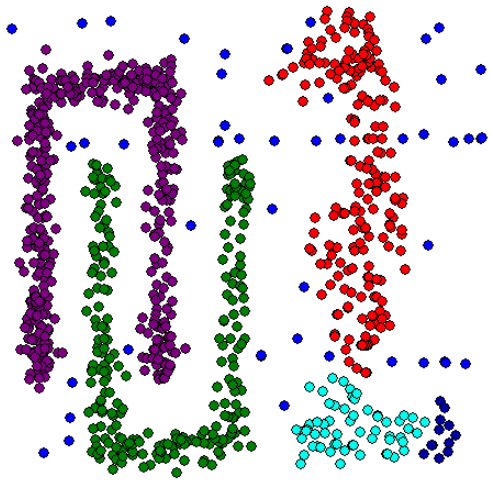


Figure 6.31: Clustering results of Shape-Outliers by DBSCAN algorithm.

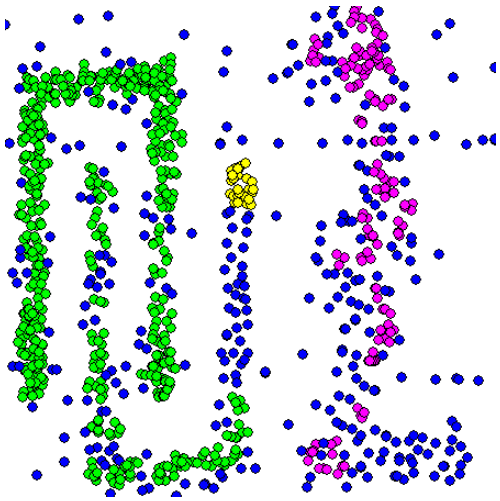


Figure 6.32: Clustering results of Shape-Outliers by CURE algorithm.

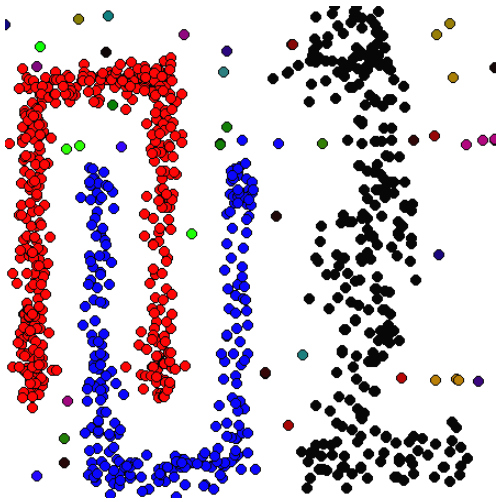


Figure 6.33: Clustering results of Shape-Outliers by CACO algorithm.

Chapter 7

Conclusions and Future Work

7.1 Summary

This thesis includes the improved algorithms for soft computing and effective and/or efficient clustering algorithms. It can be separated into five topics concerning the efficient and effective k-medoids algorithms, improved centroid-based clustering algorithms and applications, efficient Particle Swarm Optimization, efficient Ant Colony System, and improved clustering algorithm based on Ant Colony Optimization.

In Chapter 2, several efficient and effective approaches for k-medoids algorithm are proposed, such as *Clustering Large Applications based on Simulated Annealing* ($CLASA$), efficient k-medoids algorithms based on the *Partial Distance Search* (PDS), *Triangular Inequality Elimination* (TIE) and *Previous Medoid Index*. Especially, a novel approach based on the memory utilization for k-medoids algorithms are analyzed and presented.

The sampling schemes are developed for efficient and effective k-edoids algorithms in Chapter 3. The idea of the sampling schemes are motivated from the efficiency of the centroid based clustering algorithm and the higher probability of the better medoids near the centroids of the centroid based clustering so as

to develop the *Multi-Centroid, Multi-Run Sampling* scheme ($MCMRS$) and an advanced *Incremental Multi-Centroid, Multi-Run Sampling* scheme ($IMCMRS$). Experimental results demonstrate the proposed $MCMRS$ and $IMCMRS$ may dramatically reduce the computational complexity and also improve the clustering quality.

Several efficient centroid based clustering algorithms are proposed in Chapter 4. The simulated annealing is combined with the tabu search approach to get an effective clustering algorithm for image coding. Genetic clustering combined with descent algorithm is also applied for mean residual vector quantization. An incremental splitting clustering algorithm is developed for non-uniform distributed data. Especially, a novel labeled bisecting k-means clustering algorithm is proposed and applied to digital image watermarking to robust for various attacks. Several new inequalities based on the Hadamard transform are presented and applied to efficient codeword search for vector quantization.

Particle Swarm Optimization (PSO) is investigated in Chapter 5 . A parallel version for Particle Swarm Optimization termed *Parallel Particle Swarm Optimization* ($PPSO$) is proposed. Three communication strategies based on the properties of the optimization function are developed. The first communication strategy is designed for the parameters of the solution that are independent or are not much correlated such as the Rosenbrock function and Rastrigrin function. The second communication strategy can be applied to those parameters that are much correlated such as the Griewank function. In case the properties of the parameters are unknown, a third communication strategy can be used. Experimental results demonstrate the usefulness of the proposed $PPSO$ algorithm with three communication strategies.

In Chapter 6, *Ant System* (AS) and *Ant Colony System* (ACS) are studied based on traveler salesman problem. *Parallel Ant Colony System* ($PACS$) are proposed not only to reduce the computation time but also obtains a better solution. Seven communication strategies based on pheromone updating rule are

proposed for *Parallel Ant Colony System*. An adaptive Ant Colony Optimization scheme is proposed for data clustering called *Constrained Ant Colony Optimization* ($CACO$). The main idea is to apply the quadratic metric combined with the *Sum of K Nearest Neighbor Distances* ($SKNND$) to be instead of the Euclidean distance measure, adopts a constrained form of pheromone updating and reduces the search range so as to get an effective clustering algorithm comparing with the $ACODF$, $DBSCAN$ and $CURE$ data clustering algorithms.

7.2 Conclusions

7.2.1 Efficient and Effective K-medoids Algorithms

The simulated annealing (SA) is applied to generate K-medoids which is called Clustering Large Applications Based on Simulated Annealing ($CLASA$) algorithm in this thesis. The collection of the k medoids in $CLASA$ algorithm is called *state*. There are $\frac{T!}{k!(T-k)!}$ states for generating k medoids from T objects. It is possible to move from current state to any other states depending on the moving strategy. Preliminary experimental results demonstrate the $CLASA$ algorithm outperforms the $CLARA$ and $CLARANS$ algorithms.

Several efficient k-medoids approaches incorporating with previous medoid index, partial distance search (PDS), triangular inequality elimination (TIE) and utilization of memory are proposed. $CLARANS$ and $CLASA$ algorithms with previous medoid index, *TIE* and *PDS* are referred to as *CLARANS-ITP* and *CLASA-ITP*, respectively. *CLARANS* and *CLASA* algorithms combined with the proposed memory utilization technique are referred to as *CLARANS-M* and *CLASA-M*, respectively. The application of the proposed memory utilization technique, previous medoid index, *PDS* and *TIE* to *CLARANS* and *CLASA* are *CLARANS-MITP* and *CLASA-MITP*. Experimental results demonstrate the hybrid search method *CLARANS-MITP* can reduce the number of distance com-

putations from 87% to 98%. In terms of the computation time, the proposed *CLARANS-MITP* can reduce the computation time up to 96%. The *CLASA-MITP* also indicates the improvement over *CLARANS-MITP*. The proposed hybrid search techniques may also applied to the other clustering algorithms.

7.2.2 K-medoids Algorithms Based on Sampling Schemes

The drawback of the k-medoids algorithms is the computational complexity. The drawback can be overcome by applying the sampling schemes. Since the k-means algorithm can be several orders of magnitude faster than the k-medoids algorithm, we may apply the k-means algorithm several times to collect the candidate objects from the objects near the centers of clusters, then choose the medoids from the candidate objects. Based on this idea, Multi-Centroid, Multi-Run Sampling Scheme (*MCMRS*) and Incremental Multi-Centroid, Multi-Run Sampling Scheme (*IMCMRS*) are proposed. These two sampling schemes can also combine with the CLASA and the other k-medoids algorithms. Comparing with the *CLARANS* algorithm, the *IMCMRS* may reduce the computation time by a factor of 30 and also get a better average distance based on the Gauss-Markov source to generate 32 medoids. In fact, the proposed *IMCMRS* not only can reduce the average distance but also speed the clustering process for all datasets used in this thesis.

7.2.3 Centroid-Based Clustering Algorithms

A centroid-based clustering using a tabu search approach with simulated annealing is presented. The main idea of this algorithm is to use the tabu search approach to generate non-local moves for the clusters and apply the simulated annealing technique to select the current best solution, thus improving the cluster generation and reducing the distortion. In our proposed modified tabu search approach, if the distortion of the best solution of all iterations is same for some fixed number of iterations, we reset the current best solution using the best so-

lution of all iterations. Simulated annealing is used to decide which test solution is suitable to be the current best solution for generating the test solutions for next iteration. Experimental results confirm the proposed clustering approach by comparing with some existing tabu search based clustering methods.

Genetic algorithms in combination with the generalized Lloyd algorithm (GLA) are applied to the codebook design of mean-residual vector quantization (M/R VQ). The mean codebook and residual codebook are trained using GLA algorithm separately, then genetic algorithms are used to evaluate and evolve the combined mean codebook and residual codebook. Parameters setting approach is also analyzed so that the parameters are robust in the proposed algorithm. Experimental results also demonstrate the usefulness of this approach.

An innovative method, termed incremental splitting, is presented for our works on the clustering of non-uniformly distributed data. Taking the k-means method as the core, the proposed approach splits only clusters with the largest total error in each iteration. This heuristic has the effect of allocating more clusters to those regions having more sample data. Consistent experimental results reveal that our method outperforms commonly used heuristics, including random initialization, binary splitting, and pair-wise nearest neighbour.

7.2.4 Labeled Bisecting K-means Clustering

A novel VQ-based watermarking scheme using the labeled bisecting k-means clustering technique is presented in this thesis. Each cluster (or codeword) is labeled either 0 or 1 based on the labeling key before embedding. During the embedding phase, each input block is assigned to the nearest codeword or cluster centre whose label is equal to the watermark bit. The watermark extraction can be performed without the original image. Experimental results confirm the effectiveness of the proposed algorithm, and the proposed algorithm is robust to JPEG compression and some spatial-domain processing operations.

7.2.5 Hadamard Transform Based Inequalities for Efficient Clustering

An efficient nearest neighbor codeword search algorithm based on Hadamard transform for vector quantization. Four efficient elimination criteria are derived from two important inequalities based on three characteristic values in the Hadamard transform domain. Before the encoding process, all codewords in the codebook are Hadamard-transformed and sorted in the ascending order of their first elements. During the encoding process, we firstly perform the transform on the input vector and calculate its characteristic values, and initialize the current closest codeword of the input vector to be the codeword whose first element of Hadamard transform is nearest to that of the input vector, and secondly use the proposed elimination criteria to find the nearest codeword to the input vector using the up-down search mechanism near the initial best-match codeword. The main contribution for the proposed algorithm is to analyze the characteristics of the Hadamard-transform based codeword search and derive several efficient inequalities. Experimental results demonstrate the proposed algorithm is much more efficient than most existing nearest neighbor codeword search algorithms, particularly in the case of high dimension.

7.2.6 PPSO with Communication Strategies

A parallel particle swarm optimization (PPSO) algorithm is studied in this thesis. Three communication strategies for the PPSO algorithm are presented. The first communication strategy is designed for the parameters of the solution that are independent or are not much correlated such as the Rosenbrock function and Rastrigrin function. The second communication strategy can be applied to those parameters that are much correlated such as the Griewank function. In case the properties of the parameters are unknown, a third communication strategy can be used. Experimental results demonstrate the usefulness of the proposed PPSO

algorithm with three communication strategies.

7.2.7 PACS with Communication Strategies

A Parallel Ant Colony System (PACS) with communication strategies is developed. The artificial ants are partitioned into several groups. Seven communication methods for updating the pheromone level between groups in PACS are proposed and work on the traveling salesman problem using our proposed system is presented. Experimental results based on three well-known traveling salesman data sets demonstrate the proposed PACS with communication strategies are superior to the existing Ant Colony System (ACS) and Ant System (AS) with similar or better running times.

7.2.8 Constrained Ant Colony Optimization for Data Clustering

Ant colony system is discussed and extended to a novel data clustering process using Constrained Ant Colony Optimization (CACO). The CACO algorithm extends the Ant Colony Optimization algorithm by accommodating a quadratic distance metric, the Sum of K Nearest Neighbor Distances (SKNND) metric, constrained addition of pheromone and a shrinking range strategy to improve data clustering. We show that the CACO algorithm can resolve the problems of clusters with arbitrary shapes, clusters with outliers and bridges between clusters. Experimental results based on synthetic data sets demonstrate the proposed CACO algorithm can outperform the Ant Colony Optimization with Different Favor (ACODF), DBSCAN and CURE algorithms.

7.3 Future Work

7.3.1 Transform Domain Based K-medoids Algorithm

The k-medoids algorithms have been shown to be robust to outliers and are not generally influenced by the order of presentation of objects. Moreover, k-medoids algorithms are invariant to translations and orthogonal transformations of objects. Several efficient and effective approaches are developed for k-medoids algorithms, however, no one utilizes the properties of transform domain for generating medoids. Further work will develop new k-medoids algorithms based on transform domain, such as the Hadamard Transform (HT), Principal Component Transform, Discrete Cosine Transform (DCT) and Discrete Wavelet Transform (DWT). Several Hadamard Transform based inequalities developed in this thesis can combine with the k-medoids algorithms to further improve the efficiency and effectiveness.

7.3.2 PSO for Clustering of Objects

To the best of our knowledge, no one has yet applied the PSO for clustering. Further work may apply the PSO and PPSO for clustering the numerical objects. The positions of several particles may group to form a cluster. It is an open issue how to form a cluster from several particles and what are the properties to determine the total number of clusters from the history of the positions of the particles. In the beginning, the PSO and PPSO may be applied to design the centroid-based numerical clustering for vector quantization. Then further work may modify this numerical object clustering to fit the categorical objects by getting the experience from the extending the k-means to k-modes algorithm (Chiang, Chu, Hsin & Pan 2003, Huang 1998, Huang & Ng 1999, Wong & Ng 2000).

7.3.3 Sampling Scheme and Tree Structure for CACO

If the size of the data set for clustering is too big, it would be time consuming for CACO algorithm. This problem may be solved by applying the R-tree data structure and/or the development of sampling scheme for CACO algorithm. Further work will develop some new sampling schemes for CACO algorithm and apply the new mean-variance-norm pyramid (Lee & Chen 1995, Pan, Lu & Sun 2000, Song & Ra 2002*b*) search technique to R-tree data structure for CACO algorithm.

7.3.4 Application of CACO for Texture Segmentation

Texture segmentation has been shown to be an important issue and practical application in the area of image processing. The texture segmentation is to separate the whole texture into several sub-textures so that boundaries among the sub-textures best match. In fact, the texture segmentation can be formalized as the clustering problem. However no one has applied the Ant System or Ant Colony System to texture segmentation. Further work could apply the CACO for the segmentation of texture image.

7.3.5 Application of CACO for Clustering of Categorical Objects

CACO algorithm may also be applied to the clustering of categorical objects. The first step is to define the distance between different attribute values. The distance may also be obtained by getting the assistance of some optimization techniques such as the genetic algorithm (Chiang et al. 2003). Then apply the CACO algorithm to the clustering of categorical objects by modifying the quadratic distance metric, Sum of K Nearest Neighbor Distance, the visit probability of non-visited objects and the constrained addition of pheromone and shrinking range of strategy.

Appendix A

Publications arising from this thesis

Book Chapter

1. Shu-Chuan Chu, C.-S. Shieh and John F. Roddick, 2004, "A tutorial on meta-heuristics for optimization", *Intelligent Watermarking Techniques*, J. S. Pan, H. C. Huang, and L. C. Jain (eds.), Chapter 4, pp. 97-132, ISBN: 9-81238-757-9, World Scientific Publishing Company, Singapore.

Journal paper

1. Shu-Chuan Chu, John F. Roddick and Jeng-Shyang Pan, 2004, "Ant colony system with communication strategies", *Information Sciences*, ISSN: 0020-0255. (to appear)

2. Shu-Chuan Chu, John F. Roddick and Jeng-Shyang Pan, 2004, "Novel multi-centroid, multi-run sampling schemes for k-medoids-based algorithms", *International Journal of Knowledge-Based Intelligent Engineering Systems*, Vol. 8, ISSN: 1327-2314, pp. 45-56.

3. Shu-Chuan Chu, John F. Roddick and Tsong-Yi Chen, 2004, "A genetic clustering algorithm for mean-residual vector quantization", *Chinese Journal of Electronics*, Vol. 13, No. 2, ISSN: 1022-4653, pp. 316-320.

4. Shu-Chuan Chu, John F. Roddick, Zhe-Ming Lu and Jeng-Shyang Pan, January 2004, "A digital image watermarking method based on labeled bisecting clustering algorithm", *IEICE Transactions on Fundamentals of*

Electronics, Communication and Computer Sciences, Vol. E87-A, No.1, ISSN: 0916-8508, pp. 282-285.

5. Shu-Chuan Chu and John F. Roddick, 2003, "A clustering algorithm using the tabu search approach with simulated annealing for vector quantization", *Chinese Journal of Electronics*, Vol. 12, No. 3, ISSN: 1022-4653, pp. 349-353.

Conference paper

1. Jeng-Shyang Pan, Shu-Chuan Chu, John F. Roddick and Che-Zen Su, 2004, "Constrained ant colony optimization for data clustering", *The 8th Pacific Rim International Conference on Artificial Intelligence (PRICAI2004)*, Auckland, New Zealand. (to appear)

2. Shu-Chuan Chu, John F. Roddick and Jeng-Shyang Pan, 2004, "Communication strategies based particle swarm optimization algorithms", *International Workshop on Fuzzy Systems And Innovational Computing 2004 (FIC2004)*, Kitakyushu, Fukuoka, Japan. (to appear)

3. Shu-Chuan Chu, John F. Roddick, Zhe-Ming Lu and Jeng-Shyang Pan, 2004, "Hadamard transform based equal-average equal-variance equal-norm nearest neighbor codeword search algorithm", *The 2004 IEEE International Conference on Multimedia And Expo (ICME'2004)*, Taipei, Taiwan, IEEE press. (to appear)

4. Shu-Chuan Chu, John F. Roddick, Zhe-Ming Lu and Jeng-Shyang Pan, 2003, "VQ-based watermarking method using labelled bisecting k-means clustering algorithm", *The 7th International Symposium on Consumer Electronics (ISCE'03)*, Sydney, Australia, IEEE press, CD-R: ISCE–03018.

5. Shu-Chuan Chu, John F. Roddick, Jeng-Shyang Pan and Che-Jen Su, 2003, "Parallel ant colony systems", *The 14th International Symposium on Methodologies for Intelligent Systems (ISMIS 2003)*, Ning Zhong *et al.* (eds.), LNAI 2871, Springer-Verlag Heidelberg, ISSN: 0302-9743, Maebashi City, Japan, pp. 279–284.

6. Shu-Chuan Chu, John F. Roddick, Tsong-Yi Chen and Jeng-Shyang Pan, 2002, "Efficient search approaches for k-medoids-based algorithms", *The 17th IEEE Region 10 International Conference on Computers, Communications, Control and Power Engineering (IEEE TENCON'02)*, Beijing, China, IEEE Press, pp. 712a–715a.

7. Shu-Chuan Chu, John F. Roddick and Jeng-Syang Pan, 2002, "An incremental multi-centroid, multi-run sampling scheme for k-medoids-based algorithms", *The 3rd International Conference on Data Mining And Databases for Engineering, Finance And Other Fields*, A. Zanasi *et al.* (eds.), WIT Press, ISSN: 1470-6326, Bologna, Italy, pp. 553–562.

8. Shu-Chuan Chu, John F. Roddick and Jeng-Shyang Pan, 2002, "An efficient k-medoids-based algorithm using previous medoid index, triangular inequality elimination criteria and partial distance search", *The 4th International Conference on Data Warehousing and Knowledge Discovery (DaWaK 2002)*, Yahiko Kambayashi *et al.* (eds.), LNCS 2454, Springer-Verlag Heidelberg, ISSN: 0302-9743, Aix-en-Provence, France, pp. 63–72.

9. Shu-Chuan Chu, John F. Roddick and Tsong-Yi Chen, 2002, "Genetic clustering algorithm for mean-residual vector quantization", *The 18th International Conference on Advanced Science and Technology (ICAST 2002)*, Chicago, USA, ICAST press, pp. 91–98.

10. Shu-Chuan Chu, John F. Roddick and Jeng-Shyang Pan, 2002, "Efficient k-medoids algorithms using multi-centroids with multi-runs sampling scheme", *The 6th Pacific-Asia Conference on Knowledge Discovery and Data Mining in Conjunction of Workshop on Mining Data across Multiple Customer Touchpoints for CRM*, Taipei, Taiwan, pp. 14–25.

11. Shu-Chuan Chu, John F. Roddick and Jeng-Shyang Pan, 2001, "A comparative study and extension to k-medoids algorithms", *The 5th International Conference on Optimization Techniques and Applications (ICOTA 2001)*, D. Li (ed.), Hong Kong, Vol. 4, ICOTA press, pp. 1708–1717.

12. Shu-Chuan Chu and John F. Roddick, 2001, "Pattern clustering using incremental splitting for non-uniformly distributed data", *The International Conference on Knowledge-Based Intelligent Information Engineering Systems and Allied Technologies (KES 2001)*, N. Baba *et al.* (eds.), Vol. 2, IOS Press, ISBN: 1 58603 192 9, Brighton, UK, pp. 1037–1041.

13. Shu-Chuan Chu and John F. Roddick, 2000, "A clustering algorithm using the tabu search approach with simulated annealing", *The 2nd International Conference on Data Mining And Databases for Engineering, Finance And Other Fields*, N. Ebecken *et al.* (eds.), WIT Press, ISSN: 1470-6326, Cambridge, UK, pp. 515–523.

Bibliography

Abramson, D. & Abela, J. (1991), A parallel genetic algorithm for solving the school timetabling problem, Technical Report Technical Report, Division of Information Technology, CSIRO.

Agrawal, R., Gehrke, J., Gunopulos, D. & Raghavan, P. (1998), Automatic subspace clustering of high dimensional data for data mining applications, *in* 'ACM SIGMOD International Conference on the Management of Data, Seattle', pp. 94–105.

Al-Sultan, K. (1995), 'A tabu search approach to the clustering problem', *Pattern Recognition* **28**(9), 1443–1451.

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

Angeline, P. (1998), Evolutionary optimization versus particle swarm optimization: philosophy and performance differences, *in* 'Proc. Seventh Annual Conference on Evolutionary Programming', pp. 601–611.

Ankerst, M., Breunig, M. M., Kriegel, H. P. & Sander, J. (1999), OPTICS: Ordering points to identify the clustering structure, *in* 'Proc. ACM SIGMOD', pp. 49–60.

Babu, T. R. & Murty, M. N. (2001), 'Comparison of genetic algorithm based prototype selection schemes', *Pattern Recognition* **34**, 523–525.

Baek, S. J., Jeon, B. K. & Sung, K. M. (1997), 'A fast encoding algorithm for vector quantization', *IEEE Signal Processing Letters* **4**(2), 325–327.

Bei, C. D. & Gray, R. M. (1985), 'A improvement of the minimum distortion encoding algorithm for vector quantization', *IEEE Transactions on Communication* **COM-33**(10), 1132–1133.

Bland, J. A. (1999), 'Space-planning by ant colony optimization', *International Journal of Computer Applications in Technology* **12**(6), 320–328.

Brindle, A. (1981), Genetic algorithms for function optimization, PhD thesis, University of Alberta, Edmonton, Canada.

Bull, L. (2001), 'On coevolutionary genetic algorithms', *Soft Computing* **5**(3), 201–207.

Bullnheimer, B., Kotsis, G. & Strauss, C. (1997), Parallelization strategies for the ant system, Technical Report POM 9/97, Institute of Applied Computer Science, University of Vienna, Austria.

Cetin, A. & Weerackody, V. (1988), 'Design of vector quantizers using simulated annealing', *IEEE Transactions on Circuits and Systems for Video Technology* **35**(12), 1550–1555.

Chen, S. H. (2002), *AI Economics*, Talk in the Department of International Trade, National Kaohsiung University of Applied Sciences, Kaohsiung, Taiwan.

Chen, S. H. & Pan, J. S. (1989), 'Fast search algorithm for VQ-based recognition of isolated word', *IEE Proc. I* **136**(6), 391–396.

Chiang, C. S., Chu, S. C., Hsin, Y. C. & Pan, J. S. (2003), A new dissimilarity measure for k-modes algorithm, *in* '2003 International Conference on Informatics, Cybernetics and Systems', pp. 503–508.

Chu, S. C. & Fang, H. L. (1999), Genetic algorithms vs. tabu search in timetable scheduling, *in* 'Third International Conference on Knowledge-Based Intelligent Information Engineering Systems', Adelaide, Australia, pp. 492–495.

Chu, S. C. & Roddick, J. F. (2000), A clustering algorithm using the tabu search approach with simulated annealing, *in* N. Ebecken & C. A. Brebbia, eds, 'Second International Conference on Data Mining and Databases for Engineering, Finance and Other Field', Cambridge, UK, pp. 515–523.

Chu, S. C. & Roddick, J. F. (2001), Pattern clustering using incremental splitting for non-uniformly distributed data, *in* N. Baba, L. C. Jain & R. J. Howlett, eds, 'The International Conference on Knowledge-Based Intelligent Information Engineering Systems and Allied Technologies (KES2001)', Brighton, UK, pp. 1037–1041.

Chu, S. C. & Roddick, J. F. (2002), Genetic clustering algorithm for mean-residual vector quantization, *in* '18th International Conference on Advanced Science and Technology (ICAST2002)', Vol. 2, Chicago, USA, pp. 91–98.

Chu, S. C. & Roddick, J. F. (2003), 'A clustering algorithm using the tabu search approach with simulated annealing for vector quantization', *Chinese Journal of Electronics* **12**(3), 349–353.

Chu, S. C., Roddick, J. F. & Chen, T. Y. (2004), 'A genetic clustering algorithm for mean-residual vector quantization', *Chinese Journal of Electronics* **13**(2), 316–320.

Chu, S. C., Roddick, J. F., Chen, T. Y. & Pan, J. S. (2002), Efficient search approaches for k-medoids-based algorithms, *in* 'International Conference on Computers, Communications, Control and Power Engineering (IEEE TENCON'02)', IEEE Press, Beijing, China, pp. 712a–715a.

Chu, S. C., Roddick, J. F., Lu, Z. M. & Pan, J. S. (2003), VQ-based watermarking method using labelled bisecting k-means clustering algorithm, *in* '2003 IEEE International Symposium on Consumer Electronics (ISCE'03)', Sydney, Australia, pp. ISCE–03018.

Chu, S. C., Roddick, J. F., Lu, Z. M. & Pan, J. S. (2004*a*), 'A digital image watermarking method based on labeled bisecting clustering algorithm', *IEICE Transactions on Fundamentals of Electronics, Communication and Computer Sciences* **E87-A**(1), 282–285.

Chu, S. C., Roddick, J. F., Lu, Z. M. & Pan, J. S. (2004*b*), Hadamard transform based equal-average equal-variance equal-norm nearest neighbor codeword search algorithm, *in* '2004 IEEE International Conference on Multimedia And Expo (ICME'2004)', Taipei, Taiwan.

Chu, S. C., Roddick, J. F. & Pan, J. S. (2001), A comparative study and extensions to k-medoids algorithms, *in* D. Li, ed., '5th International Conference on Optimization : Techniques and Applications (ICOTA 2001)', Vol. 4, Hong Kong, pp. 1708–1717.

Chu, S. C., Roddick, J. F. & Pan, J. S. (2002*a*), Efficient k-medoids algorithms using multi-centroids with mlulti-runs sampling scheme, *in* S. Y. Hwang, J. Srivastava, J. H. Wang & E. P. Lim, eds, 'International Workshop on Mining Data across Multiple Customer Touchpoints for CRM', Taipei, Taiwan, pp. 14–25.

Chu, S. C., Roddick, J. F. & Pan, J. S. (2002*b*), An efficient k-medoids-based algorithm using previous medoid index, triangular inequality elimination criteria and partial distance search, *in* Y. Kambayashi, W. Winiwarter & M. Arikawa, eds, 'Fourth International Conference on Data Warehousing and Knowledge Discovery (DaWak 2002)', Vol. 2454 of *LNCS*, Springer-Verlag, Aix-en-Provence, France, pp. 63–72.

Chu, S. C., Roddick, J. F. & Pan, J. S. (2002*c*), An incremental multi-centroid, multi-run sampling scheme for k-medoids-based algorithms, *in* A. Zanasi, C. A. Brebbia, N. F. F. E. Ebecken & P. Melli, eds, 'Third International Conference on Data Mining and Databases for Engineering, Finance and Other Field', WIT Press, Bologna, Italy, pp. 553–562.

Chu, S. C., Roddick, J. F. & Pan, J. S. (2002*d*), An incremental multi-centroid, multi-run sampling scheme for k-medoids algorithms-extended report, Technical Report KDM-02-003, KDM Laboratory, Flinders University.

Chu, S. C., Roddick, J. F. & Pan, J. S. (2004*a*), 'Ant colony system with communication strategies', *Information Sciences* .

Chu, S. C., Roddick, J. F. & Pan, J. S. (2004*b*), Communication strategies based particle swarm optimization algorithms, *in* 'International Workshop on Fuzzy Systems and Innovational Computing 2004 (FIC2004)', Kitakyushu, Fukuoka, Japan.

Chu, S. C., Roddick, J. F. & Pan, J. S. (2004*c*), 'Novel multi-centroid, multi-run sampling schemes for k-medoids-based algorithm', *International Journal of Knowledge-Based Intelligent Engineering Systems* .

Chu, S. C., Roddick, J. F., Pan, J. S. & Su, C. J. (2003), Parallel ant colony systems, *in* N. Zhong, Z. W. *Raś*, S. Tsumoto & E. Suzuki, eds, '14th International Symposium on Methodologies for Intelligent Systems', Vol. 2871 of *LNAI*, Springer-Verlag, Maebashi City, Japan, pp. 279–284.

Clerc, M. & Kennedy, J. (2002), 'The particle swarm-explosion, stability, and convergence in a multidimensional complex space', *IEEE Transactions on Evolutionary Computation* **6**(1), 58–73.

Cohoon, J. P., Hegde, S. U., Martine, W. N. & Richards, D. (1987), Punctuated equilibria: a parallel genetic algorithm, *in* 'Second International Conference on Genetic Algorithms', pp. 148–154.

Colorni, A., Dorigo, M. & Maniezzo, V. (1991), Distributed optimization by ant colonies, *in* F. Varela & P. Bourgine, eds, 'First Europ. Conference Artificial Life', pp. 134–142.

Cox, I. J., Kilian, J., Leighton, F. T. & Shamoon, T. (1997), 'Secure spread spectrum watermarking for multimedia', *IEEE Transactions on Image Processing* **6**(12), 1673–1687.

Davis, L. (1991), *Handbook of genetic algorithms*, Van Nostrand Reinhold, New York.

Delport, V. & Koschorreck, M. (1995), 'Genetic algorithm for codebook design in vector quantization', *Electronics Letters* **31**(2), 84–85.

Dorigo, M., Caro, G. D. & Gambardella, L. M. (1999), 'Ant algorithms for discrete optimization', *Artificial Life* **5**(2), 137–172.

Dorigo, M. & Gambardella, L. M. (1997), 'Ant colony system: A cooperative learning approach to the traveling salesman problem', *IEEE Transactions on Evolutionary Computation* **26**(1), 53–66.

Dorigo, M., Maniezzo, V. & Colorni, A. (1996), 'The ant system: optimization by a colony of cooperating agents', *IEEE Transactions on Systems, Man, and Cybernetics-Part B* **26**(2), 29–41.

Dubois, D. & Prade, H. (1980), *Fuzzy Sets and Systems: Theory and Applications*, Academic Press, Orlando, FL.

Eberhart, R. & Hu, X. (1999), Human tremor analysis using particle swarm optimization, *in* 'Congress on Evolutionary Computation', pp. 1927–1930.

Eberhart, R. & Shi, Y. (1998), Comparison between genetic algorithms and particle swarm optimization, *in* 'Seventh Annual Conference on Evolutionary Programming', pp. 611–619.

Enright, A. J. & Ouzounis, C. A. (2000), 'GENERAGE: a robust algorithm for sequence clustering and domain detection', *Bioinformatics* **16**(5), 451–457.

Equitz, W. H. (1989), 'A new vector quantization clustering algorithm', *IEEE Transactions On Acoustics, Speech, and Signal Processing* **37**, 1568–1575.

Ester, M., Kriegel, H. P., Sander, J. & Xu, X. (1996), A density-based algorithm for discovering clusters in large spatial databases with noise, *in* E. Simoudis, J. Han & U. Fayyad, eds, 'Second International Conference on Knowledge Discovery and Data Mining', AAAI Press, Portland, Oregon, pp. 226–231.

Estivill-Castro, V. & Lee, I. (2000*a*), AUTOCLUST: Automatic clustering via boundary extraction for massive point-data sets, *in* 'Fifth International Conference on Geocomputation', Springer-Verlag.

Estivill-Castro, V. & Lee, I. (2000*b*), AUTOCLUST+:automatic clustering of point-data sets in the presence of obstacles, *in* J. F. Roddick & K. Hornsby, eds, 'First International Workshop on Temporal, Spatial and Spatial-Temporal Data Mining, TSDM2000', Vol. 2007 of *LNAI*, Springer-Verlag, Lyon, France, pp. 133–146.

Fisher, D. (1987), Improving inference through conceptral clustering, *in* '1987 AAAI Conference', pp. 461–465.

Fisher, D., Xu, L., Carnes, J. R., Reich, Y., Fenves, S. J., Chen, J., Shiavi, R., Biswas, G. & Weinberg, J. (1993), 'Applying ai clustering to engineering tasks', *IEEE Intelligent Systems* **8**(6), 51–60.

Flanagan, J., Morrell, D., Frost, R., Read, C. & Nelson, B. (1989), Vector quantization codebook generation using simulated annealing, *in* 'IEEE International Conference on Acoustics Speech and Signal Compression', pp. 1759–1762.

Fonseca, C. M. & Fleming, P. J. (1993), Multiobjective genetic algorithms, *in* 'IEE Colloquium on Genetic Algorithms for Control Systems Engineering', number 1993/130, pp. 6/1–6/5.

Fonseca, C. M. & Fleming, P. J. (1998), 'Multiobjective optimization and multiple constraint handling with evolutionary algorithms i: A unified formulation', *IEEE Transactions on Systems, Man and Cybernetics-Part A* **28**(1), 26–37.

Franti, P., Kivijarvi, J. & Nevalainen, O. (1998), 'Tabu search algorithm for codebook generation in vector quantization', *Pattern Recognition* **31**(8), 1139–1148.

Fukuyama, Y. & Yoshida, H. (2001), A particle swarm optimization for reactive power and voltage control in electric power systems, *in* 'Congress on Evolutionary Computation', pp. 87–93.

Gallego, R. A., Alves, A. B., Monticelli, A. & Romero, R. (1997), 'Parallel simulated annealing applied to long term transmission network expansion planning', *IEEE Transactions on Power Systems* **12**(1), 181–188.

Gamal, A., Hemachandra, L., Shperling, I. & Wei, V. (1987), 'Using simulated annealing to design good codes', *IEEE Transactions on Information Theory* **IT-33**(1), 116–123.

Ganti, V., Gehrke, J. & Ramakrishnan, R. (1999), CACTUS – clustering categorical data using summaries, *in* S. Chaudhuri & D. Madigan, eds, 'Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining', ACM Press, San Diego, CA, pp. 73–83.

Gao, Y., S. L. & Yao, P. (2000), Study on multi-objective genetic algorithm, *in* 'Proceedings of the Third World Congress on Intelligent Control and Automation', Sydney, Australia, pp. 646–650.

Gen, M. & Cheng, R. (1997), *Genetic algorithm and engineering design*, John Wiley and Sons, New York.

Gersho, A. & Gray, R. M. (1992), *Vector Quantization and Signal Compression*, Kluwer, Boston, MA.

Glover, F. (1977), 'Heuristics for integer programming using surrogate constraints', *Decision Sci.* pp. 156–166.

Glover, F. (1986), 'Future paths for integer programming and links to artificial intelligence', *Comp. Operations Res.* pp. 533–549.

Glover, F. & Laguna, M. (1997), *Tabu Search*, Kluwer Academic Publishers.

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

Guan, L. & Kamel, M. (1992), 'Equal-average hyperplane partitioning method for vector quantization of image data', *Pattern Recognition Letters* **13**(10), 693–699.

Guha, S., Rastogi, R. & Shim, K. (1998), CURE: an efficient clustering algorithm for large databases, *in* 'ACM SIGMOD International Conference on the Management of Data', Seattle, WA, USA, pp. 73–84.

Guha, S., Rastogi, R. & Shim, K. (1999), ROCK: A robust clustering algorithm for categorical attributes, *in* '1999 International Conference Data Engineering (ICDE'99)', Sydney, Australia, pp. 512–521.

Han, E. H., Karypis, G., V., K. & Mobasher, B. (1997), Clustering based on association rule hypergraphs, *in* 'SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery', Tucson, Arizona, pp. 9–13.

Han, J. & Kamber, M. (2001), *Data Mining: Concepts and Techniques*, Morgan Kaufmann, San Francisco, USA.

Han, J., Kamber, M. & Tung, A. K. H. (2001), Spatial clustering methods in data mining: A survey, *in* 'Geographic Data Mining and Knowledge Discovery', London: Taylor and Francis, Research Monographs in Geographic Information Sytems.

Handa, H., B. M. H. T. & Katai, O. (2002), 'A novel hybrid framework of coevolutionary GA and machine learning', *International Journal of Computational Intelligence and Appli-cations* **2**(1), 33–52.

Hartigan, J. A. (1975), *Clustering Algorithms*, John Wiley and Sons, Inc., New York, NY.

Hinneburg, A. & Keim, D. A. (1998), An efficient approach to clustering in large multimedia databases with noise, *in* '1998 International Conference Knowledge Discovery and Data Mining (KDD'98)', pp. 58–65.

Holland, J. (1975), *Adaptation In Natural and Artificial Systems*, University of Michigan Press.

Huang, C. M., ChenBi, Q., Stiles, G. S. & Harris, R. W. (1992), 'Fast full search equivalent encoding algorithms for image compression using vector quantization', *IEEE Transactions on Image Processing* **1**(3), 413–416.

Huang, H. C., Chu, S. C., Pan, J. S. & Lu, Z. M. (2001), 'A tabu search based maximum descent algorithm for VQ codebook design', *Journal of Information Science and Engineering* **17**, 753–762.

Huang, H. C., Pan, J. S., Lu, Z. M., Sun, S. H. & Hang, H. M. (2001), 'Vector quantization based on genetic simulated annealing', *Signal Processing* **81**(7), 1513–1523.

Huang, H. C., Wang, F. H. & Pan, J. S. (2001), 'Efficient and robust watermarking algorithm with vector quantization', *Electronics Letters* **37**(13), 826–828.

Huang, H., Wang, F. H. & Pan, J. S. (2002), 'A VQ-based robust multi-watermarking algorithm', *IEICE Transactions on Fundamentals of Electronics, Communication and Computer Sciences* **E85-A**(7), 1719–1726.

Huang, S. H. & Chen, S. H. (1990), 'Fast encoding algorithm for VQ-based image coding', *Electronics Letters* **26**(19), 1618–1619.

Huang, Z. (1998), 'Extensions to the k-means algorithm for clustering large data sets with categorical values', *Data Mining and Knowledge Discovery* **2**(3), 283–304.

Huang, Z. & Ng, M. K. (1999), 'A fuzzy k-medoids algorithm for clustering categorical data', *IEEE Transactions on Fuzzy Systems* **7**(4), 446–452.

Hwang, W. J., Jeng, S. S. & Chen, B. Y. (1997), 'Fast codeword search algorithm using wavelet transform and partial distance search techniques', *Electronics Letters* **33**(5).

Jain, A. K. & Dubes, R. C. (1988), *Algorithms for clustering data*, Prentice Hall, Englewood Cliffs, New Jersey.

Jain, A. K. & Flynn, P. J. (1996), *Image Segmentation using Clustering*, IEEE Press, Piscataway, NJ.

Jiang, S. D., Lu, Z. M. & Wang, Q. (2003), 'Fast norm-ordered codeword search algorithms for image vector quantization', *Chinese Journal of Electronics* **12**(3), 373–376.

Jo, M. & Kim, H. (2002), 'A digital image watermarking scheme based on vector quantization', *IEICE Transactions on Information and Systems* **E85-D**(6), 1054–1056.

Jolion, J. M., Meer, P. & Bataouche, S. (1991), 'Robust clustering with applications in computer vision', *IEEE Transactions on Pattern Analysis and Machine Intelligence* **13**(8), 791–802.

Karypis, G., Han, E. H. & Kumar, V. (1999), 'CHAMELEON: a hierarchical clustering algorithm using dynamic modeling', *Computer* **32**, 32–68.

Kaufman, L. & Rousseeuw, P. J. (1990), *Finding groups in data: an introduction to cluster analysis*, John Wiley and Sons, New York.

Kennedy, J. & Eberhart, R. (1997), A discrete binary version of the particle swarm algorithm, *in* 'IEEE International Conference on Computational Cybernetics and Simulation', pp. 4104–4108.

Kim, J., Krishnapuram, R. & Dave, R. N. (1996), 'A genetic algorithm for robust clustering based on a fuzzy least median of squares criterion', *Pattern Recognition Letters* **17**, 633–641.

Kirkpatrick, S., Gelatt, J. C. D. & Vecchi, M. P. (1983), 'Optimization by simulated annealing', *Science* **220**(4598), 671–680.

Kohonen, T. (1982), 'Self-organized formation of topologically correct feature maps', *Biological Cybernetics* **43**, 59–69.

Kohonen, T. (1995), *Self-organizing maps*, Springer-Verlag.

Kosko, B. (1992), *Neural Networks and Fuzzy Systems – A Dynamical Systems Approach to Machine Intelligence*, Prentice-Hall, Englewood Cliffs, NJ.

Krishnapuram, R., Joshi, A. & Yi, L. (1999), A fuzzy relative of the k-medoids algorithm with application to web document and snippet clustering, *in* 'IEEE International Fuzzy Systems Conference', Seoul, Korea, pp. 1281–1286.

Kung, S. Y. (1993), *Digital neural network*, Prentice Hall.

Kurbel, K., Schneider, B. & Singh, K. (1998), 'Solving optimization problems by parallel recombinative simulated annealing on a parallel computer-an application to standard cell placement in vlsi design', *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics* **28**(3), 454–461.

Lavretsky, E., Hovakimyan, N. & Calise, A. J. (2003), 'Upper bounds for approximation of continuous-time dynamics using delayed outputs and feedforward neural networks', *IEEE Transactions on Automatic Control* **48**(9), 1606–1610.

Lecompte, D., Kaufman, L. & Rousseeuw, P. J. (1986), 'Hierarchical cluster analysis of emotional concerns and personality characteristics in a freshman population', *Acta Psychiatrica Belgica* **86**, 324–333.

Lee, C. H. & Chen, L. H. (1994), 'Fast closest codeword search algorithm for vector quantization', *IEE Proc. Vision Image and Signal Processing* **141**(3), 143–148.

Lee, C. H. & Chen, L. H. (1995), 'A fast search algorithm for vector quantization using mean pyramids of codewords', *IEEE Transactions on Communications* **43**(2/3/4), 1697–1702.

Linde, Y., Buzo, A. & Gray, R. M. (1980), 'An algorithm for vector quantizer design', *IEEE Transactions on Communications* pp. 84–95.

Liu, C. & Wechsler, H. (2003), 'Independent component analysis of gabor features for face recognition', *IEEE Transactions on Neural Networks* **14**(4), 919–928.

Lo, S.-C. B., Huai, L. & Freedman, M. T. (2003), 'Optimization of wavelet decomposition for image compression and feature preservation', *IEEE Transactions on Medical Imaging* **22**(9), 1141–1151.

Lu, N. & Morrell, D. (1991), VQ codebook design using improved simulated annealing algorithms, *in* 'IEEE Intl. Conf. on Acoustics Speech and Signal Compression', pp. 673–676.

Lu, Z. M., Pan, J. S. & Sun, S. H. (2000*a*), 'Efficient codeword search algorithm based on hadamard transform', *Electronics Letters* **36**(16), 1364–1365.

Lu, Z. M., Pan, J. S. & Sun, S. H. (2000*b*), 'A modified tabu search algorithm for codeword index assignment', *Chinese Journal of Electronics* **9**(2), 166–168.

Lu, Z. M., Pan, J. S. & Sun, S. H. (2000*c*), 'VQ-based digital image watermarking method', *Electronics Letters* **36**(14), 1201–1202.

Lu, Z. M. & Sun, S. H. (2000), 'Digital image watermarking technique based on vector quantization', *Electronics Letters* **36**(4), 303–305.

Lu, Z. M. & Sun, S. H. (2003), 'Equal-averge equal-variance equal-norm nearest neighbour search algorithm for vector quantization', *IEICE Transactions on Information and Systems* **E-86D**(3), 660–663.

Lucasius, C. B., Dane, A. D. & Kateman, G. (1993), 'On k-medoid clustering of large data sets with the aid of a genetic algorithm: background, feasibility and comparison', *Analytica Chimica Acta* **282**, 647–669.

MacQueen, J. (1967), Some methods for classification and analysis of multivariate observations, *in* 'Fifth Berkeley symposium on mathematics, statistics and Probability', Vol. 1, pp. 281–296.

Makur, A. & Selvi, S. S. (2001), 'Variable dimension vector quantization based image watermarking', *Signal Processing* **81**(4), 889–893.

Mamdani, E. H. & Gaines, B. R. (1981), *Fuzzy Reasoning and Its Applications*, Academic Press, Orlando, FL.

Maniezzo, V. & Colorni, A. (1999), 'The ant system applied to the quadratic assignment problem', *IEEE Transactions on Knowledge and Data Engineering* **11**(5), 769–778.

Markl, V., Ramsak, F. & Bayer, R. (1999), Improving olap performance by multidimensional hierarchical clustering, *in* 'Proc. IDEAS Conference', Montreal, Canada.

Mobasher, B., Cooley, R. & Srivastava, J. (1999), Creating adaptive web sites through usage-based clustering of urls, *in* 'Knowledge and Data Engineering Workshop'.

Ng, B. W. & Bouzerdoun, A. (2001), k-means clustering applied to texture segmentation with complex wavelet features, *in* D. Li, ed., 'Fifth International Conference on Optimization : Techniques and Applications (ICOTA 2001)', Vol. 4, Hong Kong, pp. 1732–1739.

Ng, R. T. (1996), Spatial data mining: Discovering knowledge of clusters from maps, *in* 'ACM SIGMOD Workshop on Research Issue on Data Mining and Knowledge Discovery', Montreal, Canada.

Ng, R. T. & Han, J. (1994), Efficient and effective clustering methods for spatial data mining, *in* J. B. Bocca, M. Jarke & C. Zaniolo, eds, 'Twentieth International Conference on Very Large Data Bases', Morgan Kaufmann, Santiago, Chile, pp. 144–155.

Ng, R. T. & Han, J. (2002), 'Clarans: A method for clustering objects for spatical data mining', *IEEE Transactions on Knowledge and Data Engineering* **14**(5), 1003–1016.

O'Ruanaidh, J. J. K., Dowling, W. J. & Boland, F. M. (1996), 'Watermarking digital images for copyright protection', *IEE Proceedings-Version, Image and Signal Processing* **143**(4), 250–256.

Palicot, J. & Roland, C. (2003), 'A new concept for wireless reconfigurable receivers', *IEEE Communications Magazine* **41**(7), 124–132.

Pan, J. S. & Chu, S. C. (1996), 'Non-redundant VQ channel coding using tabu search strategy', *IEE Electronic Letters* **32**(17), 1545–1546.

Pan, J. S., Chu, S. C., Roddick, J. F. & Su, C. J. (2004), Constrained ant colony optimization for data clustering, *in* C. Zhang, H. W. Guesgen & W. K. Yeap, eds, 'Proceedings of 8th Pacific Rim International Conference on Artificial Intelligence', Springer-Verlag, Auckland, New Zealand.

Pan, J. S. & Huang, K. C. (1998), 'A new vector quantization imge coding algorithm based on the extension of the bound for minkowski metric', *Pattern Recognition* **31**(11), 1757–1760.

Pan, J. S., Lu, Z. M., Shieh, C. S. & Sun, S. H. (2000), 'Application of tabu search approach to energy allocation for index transmission of codeword', *Chinese Journal of Electronics* **9**(4), 471–474.

Pan, J. S., Lu, Z. M. & Sun, S. H. (2000), 'A fast codeword search algorithm for image coding based on mean-variance pyramids of codewords', *Electronics Letters* **36**(3), 210–211.

Pan, J. S., Lu, Z. M. & Sun, S. H. (2003), 'An efficient encoding algorithm for vector quantization based on subvector technique', *IEEE Transactions on Image Processing* **12**(3), 265–270.

Pan, J. S., McInnes, F. R. & Jack, M. A. (1995), 'VQ codebook design using genetic algorithms', *Electronics Letters* **31**(17), 1418–1419.

Pan, J. S., McInnes, F. R. & Jack, M. A. (1996*a*), 'Application of parallel genetic algorithm and property of multiple global optima to VQ codevector index assignment for noisy channels', *Electronics Letters* **32**(4), 296–297.

Pan, J. S., McInnes, F. R. & Jack, M. A. (1996*b*), 'Bound for minkowski metric or quadratic metric applied to VQ codeword search', *IEE Proc. Vision Image and Signal Processing* **143**(1), 67–71.

Pan, J. S., McInnes, F. R. & Jack, M. A. (1996*c*), 'Fast clustering algorithm for vector quantization', *Pattern Recognition* **29**(3), 511–518.

Pan, J. S., Wang, J., Fang, H. L. & Chen, C. (1998), A modifed tabu search approach for texture segmentation using $2-d$ non-separable wavelet frames, *in* '10th International Conference on Tools with Artificial Intelligence', pp. 474–481.

Pan, J. S. & Wang, J. W. (1999), 'Texture segmentation using separable and non-separable wavelet frames', *IEICE Transactions on Fundamentals of Electronics, Communication and Computer Sciences* **E82-A**(8), 1463–1474.

Papadimitriou, C. H. & Steiglitz, K. (1982), *Combinatorial Opti-mization – Algorithms and Complexity*, Prentice Hall.

Park, J. W., Harley, R. G. & Venayagamoorthy, G. K. (2003), 'Adaptive-critic-based optimal neurocontrol for synchronous generators in a power system using MLP/RBF neural networks', *IEEE Transactions on Industry Applications* **39**(5), 1529–1540.

Parpinelli, R. S., Lopes, H. S. & Freitas, A. A. (2002), 'Data mining with an ant colony optimization algorithm', *IEEE Trans. on Evolutionary Computation* **6**(4), 321–332.

Pereira, S. & Pun, T. (2000), 'An iterative template matching algorithm using the chirp-z transform for digital image watermarking', *Pattern Recognition* **33**(1), 173–175.

Ra, S. W. & Kim, J. K. (1993), 'Fast mean-distance-ordered partial codebook search algorithm for image vector quantization', *IEEE Transactions on Circuits System II* **40**(9), 576–579.

Rasmussen, E. (1992), *Clustering Algorithms. In Information Retrieval: Data Structures and Algorithms*, Prentice-Hall, Inc., Upper Saddle River.

Robinson, J. & Kecman, V. (2003), 'Combining support vector machine learning with the discrete cosine transform in image compression', *IEEE Transactions on Neural Networks* **14**(4), 950–958.

Sander, J., Ester, M., Kriegel, H.-P. & Xu, X. (1998), 'Density-based clustering in spatial databases: The algorithm GDBSCAN and its applications', *Data Mining and Knowledge Discovery* **2**(2), 169–194.

Sareni, B. & Krahenbuhl, L. (1998), 'Fitness sharing and niching methods revisited', *IEEE Transactions on Evolutionary Computation* **2**(3).

Sheikholeslami, G., Chatterjee, S. & Zhang, A. (1998), WaveCluster: A multiresolution clustering approach for very large spatial databases, *in* '1998 International Conference Very Large Data Bases (VLDB'98)', New York, pp. 428–439.

Shi, Y. & Eberhart, R. (1998), A modified particle swarm optimizer, *in* 'IEEE World Congress on Computational Intelligence', pp. 69–73.

Shi, Y. & Eberhart, R. (1999), Empirical study of particle swarm optimization, *in* 'Congress on Evolutionary Computation', pp. 1945–1950.

Shi, Y. & Eberhart, R. (2001), Fuzzy adaptive particle swarm optimization, *in* 'Congress on Evolutionary Computation', pp. 101–106.

Shi, Y. & Krohling, R. A. (2002), Co-evolutionary particle swarm optimization to solve min-max problems, *in* 'Proceedings of 2002 Congress on Evolutionary Computation CEC'2002', Vol. 2, pp. 1682–1687.

Skorin-Kapov, J. (1990), 'Tabu search applied to the quadratic assignment problem', *ORSA Journal of Computing* **2**(1), 33–45.

Soleymani, M. R. & Morgera, S. D. (1987), A heigh-speed algorithm for vector quantization, pp. 1946–1948.

Song, B. C. & Ra, J. B. (2002*a*), 'A fast search algorithm for vector quantization using L2-norm pyramid of codewords', *IEEE Transactions on Image Processing* **11**(1).

Song, B. C. & Ra, J. B. (2002*b*), 'A fast search algorithm for vector quantization using L_2-norm pyramid of codewords', *IEEE Transactions on Image Processing* **11**(1), 10–15.

Stützle, T. (1998), Parallelization strategies for ant colony optimization, *in* 'Fifth International Conference on Parallel Problem Soving for Nature', Vol. 1498, LNCS, Springer-Verlag, pp. 722–731.

Su, M. C. & Chou, C. H. (2001), 'A modified version of the k-means algorithm with a distance based on cluster symmetry', *IEEE Transactions on Pattern Analysis and Machine Intelligence* **23**(6), 674–680.

Swanson, M. D., Bin, Z. & Tewfik, A. H. (1998), 'Multiresolution scene-based video watermarking using perceptual models', *IEEE Journal on Selected Areas in Communications* **16**(4), 540–550.

Togai, M. & Watanabe, H. (1986), 'Expert system on a chip: An engine for real-time approximate reasoning', *IEEE Expert* **1**(3).

Tsai, C. F., Wu, H. C. & Tsai, C. W. (2002), A new data clustering approach for data mining in large databases, *in* 'International Symposium on Parallel Architectures, Algorithms and Networks', IEEE Press, pp. 278–283.

Vaisey, J. & Gersho, A. (1988), Simulated annealing and codebook design, *in* 'IEEE Intl. Conf. on Acoustics Speech and Signal Compression', pp. 1176–1179.

Vecchi, M. P. & Kirkpatrick, S. (1983), 'Global wiring by simulated annealing', *IEEE Transactions on Computer-Aided Design* **CAD-2**(4), 215–222.

Vidal, E. (1986), 'An algorithm for finding nearest neighbours in (approximately) constant average time', *Pattern Recognition Letters* **4**, 145–157.

Voyatzis, G. & Pitas, I. (1999), 'The use of watermarks in the protection of digital multimedia products', *Proceedings of the IEEE* **87**(7), 1197–1207.

Wang, L. & Wu, Q. (2001), Ant system algorithm for optimization in continuous space, *in* 'IEEE International Conference on Control Applications CCA'2001', pp. 395–400.

Wang, W., Yang, J. & Muntz, R. (1997), STING: A statistical information grid approach to spatial data mining, *in* 'International Conference on Very Large Data Bases', Athens, Greece, pp. 186–195.

Wang, Y., Doherty, J. F. & Van Dyck, R. (2002), 'A wavelet-based watermarking algorithm for ownership verification of digital images', *IEEE Transactions on Image Processing* **11**(2), 77–88.

Whitley, D. (1993), A genetic algorithm tutorial, Technical Report Technical Report CS-93-103, Department of Computer Science, Colorado State University, Fort Collins, CO 8052.

Wong, J. C. & Ng, M. K. (2000), A tabu search based algorithm for clustering categorical data sets, *in* 'Second International Conference on Intelligent Data Engineering and Automated Learning', LNCS, Springer-Verlag, Shatin, Hong Kong, China, pp. 559–564.

Wu, K. S. & Lin, J. C. (2000), 'Fast VQ encoding by an efficient kick-out condition', *IEEE Transactions on Circuits Systems for Video Technology* **10**(1), 59–62.

Xiang, B. & Berger, T. (2003), 'Efficient text-independent speaker verification with structural gaussian mixture models and neural network', *IEEE Transactions on Speech and Audio Processing* **11**(5), 447–456.

Yamakawa, T. (1988), Fuzzy microprocessor- rule chip and defuzzification chip, *in* 'International Workshop on Fuzzy Systems Applications', Iizuka-88, Kyushu Institute of Technology, pp. 51–52.

Zadeh, L. A. (1965), 'Fuzzy sets', *Information and Control* **8**, 338–353.

Zaïane, O. R. & Lee, C. H. (2002), Clustering spatial data when facing physical constraints, *in* 'IEEE International Conference on Data Mining (ICDM 2002)', Maebashi City, Japan, IEEE Computer Society, pp. 737–740.

Zamir, O., Etzion, O., Mandani, O. & Karp, R. (1997), Fast and intuitive clustering of web documents, *in* D. Heckerman, H. Mannila, D. Pregibon & R. Uthurusamy, eds, 'Third International Conference on Knowledge Discovery and Data Mining', AAAI Press, Menlo Park, California, Newport Beach, CA, USA, pp. 287–290.

Zeger, K. A. & Gersho, A. (1989), 'Stochastic relatxation algorithm for improved vector quantizer design', *Electronics Letters* **25**(14), 896–898.

Zeger, K. A., Vaisey, J. & Gersho, A. (1992), 'Globally optimal vector quantizer design by stochastic relaxation', *IEEE Transactions on Signal Processing* **40**(2), 310–322.

Zhang, T., Ramakrishnan, R. & Livny, M. (1996), BIRCH: An efficient clustering method for very large databases, *in* 'ACM SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery', Montreal, Canada, pp. 103–114.