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
© (Shu-Chuan Chu, 2004)
Contents

Abstract xiv
Certification xv
Acknowledgements xvi

1 Introduction 1
  1.1 Related Existing Clustering Algorithms . . . . . . . . . . . . . . 5
  1.2 Soft Computing . . . . . . . . . . . . . . . . . . . . . . . . . . . 8
    1.2.1 Fuzzy Theorem . . . . . . . . . . . . . . . . . . . . . . . . 8
    1.2.2 Artificial Neural Networks . . . . . . . . . . . . . . . . . 10
    1.2.3 Simulated Annealing . . . . . . . . . . . . . . . . . . . . . 11
    1.2.4 Tabu Search Approach . . . . . . . . . . . . . . . . . . . . 12
    1.2.5 Genetic Algorithms . . . . . . . . . . . . . . . . . . . . . . 12
    1.2.6 Particle Swarm Optimization . . . . . . . . . . . . . . . . 14
    1.2.7 Ant Systems and Ant Colony Systems . . . . . . . . . . . 15
  1.3 Thesis Structure . . . . . . . . . . . . . . . . . . . . . . . . . . 16

2 Improved $K$-medoids Algorithms 19
  2.1 Related existing $K$-Medoids Algorithms . . . . . . . . . . . . 19
    2.1.1 $PAM$ - Partitioning Around Medoids . . . . . . . . . . . 21
    2.1.2 $CLARA$ – Clustering LARge Applications . . . . . . . . 22
    2.1.3 $CLARANS$ – Clustering Large Applications Based on Ran-
        domized Search . . . . . . . . . . . . . . . . . . . . . . . . . 24
    2.1.4 Fuzzy $K$-Medoids Algorithms . . . . . . . . . . . . . . . 26
    2.1.5 Genetic $K$-Medoids Algorithm . . . . . . . . . . . . . . . 28
CONTENTS

2.2 CLASA - Clustering Large Applications Based on Simulated Annealing ........................................ 28
  2.2.1 What’s CLASA ........................................ 29
  2.2.2 Experiments ........................................ 31

2.3 Efficient Search Based K-Medoids Algorithms ................. 35
  2.3.1 Revisiting Swap-Comparison of PAM ..................... 37
  2.3.2 VQ-Based Techniques .................................. 41
  2.3.3 Partial Distance Search ................................. 43
  2.3.4 Triangular Inequality Elimination ....................... 44
  2.3.5 Previous Medoid Index ................................. 48

2.4 Memory Utilization Based K-Medoids Algorithm ............... 50

2.5 Experimental Results ........................................ 52
  2.5.1 Gaussian Source ....................................... 53
  2.5.2 Gauss-Markov Source ................................... 54
  2.5.3 Rectangular Clusters ................................... 55
  2.5.4 Elliptic Clusters ...................................... 56
  2.5.5 Curved Clusters ...................................... 57
  2.5.6 Lena Dataset .......................................... 57
  2.5.7 Real-World Dataset .................................... 58
  2.5.8 Summary .............................................. 59

3 Sampling Schemes for K-Medoids Algorithm .................. 74
  3.1 Introduction .............................................. 75
  3.2 Multi-Centroid, Multi-Run Sampling Scheme (MCMRS) ...... 76
    3.2.1 Motivation .......................................... 76
    3.2.2 Experimental Results ................................ 78
  3.3 Incremental Multi-Centroid, Multi-Run Sampling Scheme (IMCMRS) 84
    3.3.1 Motivation .......................................... 84
    3.3.2 Experimental Results ................................ 85
CONTENTS

4 Improved Centroid-Based Clustering Algorithms 94
  4.1 Related Existing Centroid-Based Clustering Algorithm .......... 95
    4.1.1 K-Means (GLA) ......................................... 96
    4.1.2 Simulated Annealing for Clustering .......................... 97
    4.1.3 Tabu Search Approach for Clustering ......................... 98
    4.1.4 Clustering Using Stochastic Relaxation Approach .......... 101
  4.2 Clustering using Tabu Search with Simulated Annealing .......... 102
    4.2.1 Motivation ............................................. 102
    4.2.2 Experimental Results .................................. 103
  4.3 Genetic Clustering for Mean-Residual Vector Quantization ......... 111
    4.3.1 Motivation ............................................. 111
    4.3.2 Experiments .......................................... 118
  4.4 Incremental Splitting Clustering ................................ 125
    4.4.1 Introduction .......................................... 125
    4.4.2 Related Works ........................................ 127
    4.4.3 Proposed Algorithm – Incremental Splitting ............... 128
    4.4.4 Experimental Results .................................. 129
  4.5 Labelled Bisecting K-Means Clustering Algorithm for Watermarking 132
    4.5.1 Introduction .......................................... 132
    4.5.2 Proposed Algorithm ................................... 133
    4.5.3 Experimental Results .................................. 138
  4.6 Hadamard Transform Based Fast Codeword Search Algorithm for High-Dimensional VQ Encoding ........................................ 143
    4.6.1 Introduction .......................................... 144
    4.6.2 Related Existing Nearest Neighbour Codeword Search Algorithms ........................................ 146
    4.6.3 Basic Definitions And Properties ........................................ 153
    4.6.4 Proposed Algorithm ................................... 156
    4.6.5 Experimental Results .................................. 160

5 Parallel Particle Swarm Optimization 162
  5.1 History of Particle Swarm Optimization .......................... 162
  5.2 Particle Swarm Optimization with Communication Strategies ..... 167
    5.2.1 Motivation and Description ................................ 167
    5.2.2 Experiments .......................................... 173
## 6 Parallel and Constrained Ant Colony Optimizations

6.1 History of Ant System and Ant Colony System .......................... 178
6.2 Ant Colony System with Communication Strategies ................. 183
   6.2.1 Description ..................................................................... 183
   6.2.2 Experimental Results .................................................. 188
6.3 Adaptive Ant Colony System for Data Clustering ................. 195
   6.3.1 Ant Colony Optimization with Different Favor (ACODF) .... 195
   6.3.2 The Constrained Ant Colony Optimization (CACO) ......... 196
   6.3.3 Experiments and Results ............................................. 201

## 7 Conclusions and Future Work

7.1 Summary .............................................................................. 212
7.2 Conclusions ......................................................................... 214
   7.2.1 Efficient and Effective K-medoids Algorithms .............. 214
   7.2.2 K-medoids Algorithms Based on Sampling Schemes ... 215
   7.2.3 Centroid-Based Clustering Algorithms ....................... 215
   7.2.4 Labeled Bisecting K-means Clustering ......................... 216
   7.2.5 Hadamard Transform Based Inequalities for Efficient Clus-
tering .......................................................... 217
   7.2.6 PPSO with Communication Strategies .......................... 217
   7.2.7 PACS with Communication Strategies ......................... 218
   7.2.8 Constrained Ant Colony Optimization for Data Clustering 218
7.3 Future Work .......................................................................... 219
   7.3.1 Transform Domain Based K-medoids Algorithm ........... 219
   7.3.2 PSO for Clustering of Objects ...................................... 219
   7.3.3 Sampling Scheme and Tree Structure for CACO .......... 220
   7.3.4 Application of CACO for Texture Segmentation .......... 220
   7.3.5 Application of CACO for Clustering of Categorical Objects 220

A Publications arising from this thesis ........................................ 221

Bibliography ............................................................................. 224
List of Figures

1.1 Problem datasets for some clustering algorithms .......................... 3
1.2 The hybridization of soft computing ........................................... 9

2.1 Clustering LARge Applications (CLARA) ................................. 22
2.2 Clustering Large Applications Based on RANdomized Search Algorithm (CLARANS) ................................................................. 25
2.3 Clustering Large Applications Based on Simulated Annealing Algorithm (CLASA) ................................................................. 30
2.4 Data Generation Program ............................................................. 32
2.5 Elliptic Clusters ........................................................................ 33
2.6 Performance Comparison of CLARA, CLARANS and CLASA for Gauss-Markov Sources ................................................................. 36
2.7 Performance Comparison of CLARA, CLARANS and CLASA for Elliptic Clusters ................................................................. 36
2.8 Four Cases When Medoid \( o_{old} \) Is Replaced as Representative Object by \( o_{new} \) ................................................................. 39
2.9 Fifth Case When Medoid \( o_{old} \) Becomes The Non-Medoid Object . 40
2.10 The First Criterion of TIE ......................................................... 45
2.11 The Second Criterion of TIE ....................................................... 46
2.12 The Third Criterion of TIE ........................................................... 47
2.13 The Combination of Criterion 2 and 3 of TIE ................................. 49
2.14 The Combination of Criterion 1, 2 and 3 of TIE ............................. 49
2.15 The Three Categories of Distance Calculation ................................. 50
2.16 Usage of Memory for Distance Calculations ................................ 51
2.17 Twelve Elliptic Clusters ............................................................... 56
2.18 Curved Clusters ........................................................................ 57
2.19 Results of Gauss-Markov Experiment ........................................... 70
LIST OF FIGURES

2.20 Results of Elliptic Clusters Experiment ......................... 70
2.21 Results of Curved Clusters Experiment .......................... 71
2.22 Results of Experiments by Number of Calculations ............ 72
2.23 Results of Experiments by Number of Calculations ............ 73

3.1 Compact Clusters with Noise ........................................ 78
3.2 Performance comparison of CLARA, CLARANS, CLASA, MCMRS-
CLASA and MCMRS for Four Elliptic Clusters ...................... 80
3.3 Performance Comparison of CLARA, CLARANS, MCMRS and
MCMRS-CLASA for Twelve Elliptic Clusters ........................ 81
3.4 Performance Comparison of CLARA, CLARANS, MCMRS and
MCMRS-CLASA for Compact Clusters with Noise ........................ 82
3.5 The flowchart of Incremental Multi-Centroid, Multi-Run Sampling
Scheme (IMCMRS) ...................................................... 86
3.6 Performance Comparison of CLARA, CLARANS, MCMRS and
IMCMRS for Four Elliptic Clusters ................................. 88
3.7 Performance Comparison of CLARA, CLARANS, MCMRS and
IMCMRS for Twelve Elliptic Clusters ............................... 89
3.8 Performance comparison of CLARA, CLARANS, MCMRS and
IMCMRS for Five Compact Clusters ................................. 90
3.9 Performance Comparison of CLARA, CLARANS, MCMRS and
IMCMRS for Gauss-Markov Source ................................. 92
3.10 Performance Comparison of CLARA, CLARANS, MCMRS and
IMCMRS for Lena Image .............................................. 93

4.1 Flowchart of The Tabu Search Approach for Clustering Patterns . 99
4.2 Flowchart of The Tabu Search Approach with GLA Algorithm ... 100
4.3 Flowchart of The Tabu Search Approach with Simulated Annealing 104
4.4 Performance Comparison Using LENA Image ...................... 107
4.5 Performance Comparison Using Baboo Image ..................... 107
4.6 Performance Comparison Using Pepper Image ..................... 107
4.7 Comparison of Lena Image Recovered by Different Algorithm .... 108
4.8 Comparison of Baboo Image Recovered by Different Algorithm .... 109
4.9 Comparison of Pepper Image Recovered by Different Algorithm .... 110
4.10 Block Diagram of VQ Compression ................................. 112
4.11 Schematic for Mean-Residual Vector Quantization (M/R VQ) .... 114
### LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure No.</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.12</td>
<td>Flowchart of Genetic Clustering Algorithm for ((M/R \ VQ))</td>
<td>117</td>
</tr>
<tr>
<td>4.13</td>
<td>Experiment for Mutation Rate</td>
<td>118</td>
</tr>
<tr>
<td>4.14</td>
<td>Experiment for Survival Rate</td>
<td>119</td>
</tr>
<tr>
<td>4.15</td>
<td>Experiment for Population Size</td>
<td>119</td>
</tr>
<tr>
<td>4.16</td>
<td>Experiment for Maximum Number of Generation</td>
<td>120</td>
</tr>
<tr>
<td>4.17</td>
<td>Experiment for the Number of Iteration for GLA</td>
<td>121</td>
</tr>
<tr>
<td>4.18</td>
<td>Pseudo Code for Local-Descent Methods</td>
<td>127</td>
</tr>
<tr>
<td>4.20</td>
<td>Pseudo Code for Incremental Splitting Algorithm</td>
<td>129</td>
</tr>
<tr>
<td>4.21</td>
<td>Distribution of 10,000 Sample Data</td>
<td>130</td>
</tr>
<tr>
<td>4.22</td>
<td>Distribution of 256 Clusters Using Different Heuristics</td>
<td>131</td>
</tr>
<tr>
<td>4.23</td>
<td>A Concrete Example to Describe The Embedding Process for Each Input Vector</td>
<td>135</td>
</tr>
<tr>
<td>4.24</td>
<td>Flowchart of labelled bisecting k-means clustering algorithm</td>
<td>136</td>
</tr>
<tr>
<td>4.25</td>
<td>Original Image and Watermark</td>
<td>139</td>
</tr>
<tr>
<td>4.26</td>
<td>JPEG Compressed Watermarked Image and Corresponding Extracted watermark</td>
<td>140</td>
</tr>
<tr>
<td>4.27</td>
<td>Median Filtered Watermarked Image and Corresponding Extracted Watermark</td>
<td>141</td>
</tr>
<tr>
<td>4.28</td>
<td>Blurred Watermarked Image and Corresponding Extracted Watermark</td>
<td>141</td>
</tr>
<tr>
<td>4.29</td>
<td>Sharpened Watermarked Image and Corresponding Extracted Watermark</td>
<td>142</td>
</tr>
<tr>
<td>4.30</td>
<td>Attacked Watermarked Image by Cropping and Corresponding Extracted Watermark</td>
<td>142</td>
</tr>
<tr>
<td>5.1</td>
<td>Object function (F_2)</td>
<td>165</td>
</tr>
<tr>
<td>5.2</td>
<td>Progress of PSO on object function (F_2)</td>
<td>165</td>
</tr>
<tr>
<td>5.3</td>
<td>The distribution of particles at different iterations</td>
<td>166</td>
</tr>
<tr>
<td>5.4</td>
<td>Communication Strategy for Loosely Correlated Parameters</td>
<td>170</td>
</tr>
<tr>
<td>5.5</td>
<td>Communication Strategy for strongly correlated parameters</td>
<td>171</td>
</tr>
<tr>
<td>5.6</td>
<td>A General Communication Strategy for Unknown Correlation Between Parameters</td>
<td>172</td>
</tr>
<tr>
<td>6.1</td>
<td>A traveling salesman problem with 12 cities</td>
<td>179</td>
</tr>
</tbody>
</table>
6.2 The shortest route found by the ant system. .................. 181
6.3 The snapshot of pheromone intensities after 20 episodes. ..... 181
6.4 Update the pheromone level according to the best route of all groups ........ 185
6.5 Update the pheromone level between each pair of groups ....... 186
6.6 Update the pheromone level according to the ring structure .... 186
6.7 Update the Pheromone level to the neighbours according to the group number \( j \) differs by one bit .................... 187
6.8 Performance comparison among \( AS, ACS \) and two arbitrarily chosen strategies for EIL101 data set ......................... 189
6.9 Performance comparison among \( AS, ACS \) and two arbitrarily chosen strategies for ST70 data set ............................ 190
6.10 Performance comparison among \( AS, ACS \) and two arbitrarily chosen strategies for TSP225 data set ......................... 190
6.11 Ant tends moving toward the object with dense cluster ...... 197
6.12 Conventional search route using Equation 6.1 ............... 199
6.13 Shrinking search route using Equation 6.14 .................. 200
6.14 Clustering result of CACO (\( N_1 = \frac{1}{55} \)) ............... 201
6.15 Clustering result of CACO (\( N_1 = \frac{1}{20} \)) ............... 202
6.16 Clustering result of CACO (\( \gamma = 1 \)) .................. 202
6.17 Clustering result of CACO (\( \gamma = 5 \)) .................. 203
6.18 Clustering results of Four-Cluster by ACODF algorithm .... 204
6.19 Clustering results of Four-Cluster by CACO algorithm .... 204
6.20 Clustering results of Four-Cluster by DBSCAN algorithm .... 205
6.21 Clustering results of Four-Cluster by CURE algorithm .... 205
6.22 Clustering results of Four-Bridge by ACODF algorithm .... 206
6.23 Clustering results of Four-Bridge by DBSCAN algorithm .... 206
6.24 Clustering results of Four-Bridge by CURE algorithm .... 207
6.25 Clustering results of Four-Bridge by CACO algorithm .... 207
6.26 Clustering results of Smile-Face by ACODF algorithm .... 208
6.27 Clustering results of Smile-Face by DBSCAN algorithm .... 208
6.28 Clustering results of Smile-Face by CURE algorithm .... 209
6.29 Clustering results of Smile-Face by CACO algorithm .... 209
6.30 Clustering results of Shape-Outliers by ACODF algorithm .... 210
LIST OF FIGURES

6.31 Clustering results of Shape-Outliers by DBSCAN algorithm. . . . 210
6.32 Clustering results of Shape-Outliers by CURE algorithm. . . . . . 211
6.33 Clustering results of Shape-Outliers by CACO algorithm. . . . . . 211
List of Tables

2.1 Results of Experiment for Gaussian Source .......................... 34
2.2 Results of Experiment for Rectangular Clusters .................... 35
2.3 Results of Experiment for Gauss-Markov Source ................... 37
2.4 Results of Experiment for Elliptic Clusters ......................... 38
2.5 Summarisation of Cases Regarding Cost of Replacing $o_{old}$ with $o_{new}$ as medoid .................................................. 39
2.6 Results of Experiment for Gaussian Source .......................... 60
2.7 Results of Experiment for Gauss-Markov Source ................... 61
2.8 Results of Experiment for Gauss-Markov Source–Compared with CLARANS and Extended CLASA Algorithms ...................... 62
2.9 Results of Experiment for Rectangular Clusters ..................... 63
2.10 Results of Experiment for Rectangular Clusters–Compared with CLARANS and Extended CLASA Algorithms ..................... 64
2.11 Results of Experiment for Elliptic Clusters .......................... 65
2.12 Results of Experiment for Curved Clusters .......................... 66
2.13 Experimental Results for Ten Runs of CLARANS, CLARANS-ITP, CLARANS-M and CLARANS-MITP Algorithms for 8 Medoids of Lena Dataset ............................................ 67
2.14 Experimental Results for Ten Runs of CLARANS, CLARANS-ITP, CLARANS-M and CLARANS-MITP Algorithms for 16 Medoids of Lena Dataset ............................................ 68
2.15 Results of Experiment for Lena Dataset ............................. 68
2.16 Results of Experiment for Co-Occurrence Texture of Corel Image Collection ......................................................... 69
3.1 Results of Experiment for Four Elliptic Clusters ..................... 79
3.2 Results of Experiment for Twelve Elliptic Clusters .................. 81
3.3 Results of Experiment for Compact Clusters .......................... 82
### LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.4</td>
<td>Results of Experiment for Lena Image</td>
<td>83</td>
</tr>
<tr>
<td>3.5</td>
<td>Results of Experiment for Four Elliptic Clusters</td>
<td>88</td>
</tr>
<tr>
<td>3.6</td>
<td>Results of Experiment for Twelve Elliptic Clusters</td>
<td>89</td>
</tr>
<tr>
<td>3.7</td>
<td>Results of Experiment for Compact Clusters</td>
<td>90</td>
</tr>
<tr>
<td>3.8</td>
<td>Results of Experiment for Gauss-Markov Source</td>
<td>91</td>
</tr>
<tr>
<td>3.9</td>
<td>Results of Experiment for Lena Image</td>
<td>93</td>
</tr>
<tr>
<td>4.1</td>
<td>Performance Comparison Using Lena Image as The Training Patterns</td>
<td>105</td>
</tr>
<tr>
<td>4.2</td>
<td>Performance Comparison Using Baboo Image as The Training Patterns</td>
<td>106</td>
</tr>
<tr>
<td>4.3</td>
<td>Performance Comparison Using Pepper Image as The Training Pattern</td>
<td>106</td>
</tr>
<tr>
<td>4.4</td>
<td>Performance Comparison for M/R VQ with Mean Codebook Size 64 and Residual Codebook Size 64 Using Pepper Image</td>
<td>121</td>
</tr>
<tr>
<td>4.5</td>
<td>Performance Comparison for M/R VQ with Mean Codebook Size 64 and Residual Codebook Size 64 Using Lena Image</td>
<td>122</td>
</tr>
<tr>
<td>4.6</td>
<td>Performance Comparison for M/R VQ with Mean Codebook Size 128 and Residual Codebook Size 128 Using Pepper Image</td>
<td>122</td>
</tr>
<tr>
<td>4.7</td>
<td>Performance Comparison for M/R VQ with Mean Codebook Size 128 and Residual Codebook Size 128 Using Lena Image</td>
<td>123</td>
</tr>
<tr>
<td>4.8</td>
<td>Performance Comparison for M/R VQ with mean Codebook Size 256 and Residual Codebook Size 256 Using Pepper Image</td>
<td>123</td>
</tr>
<tr>
<td>4.9</td>
<td>Performance Comparison for M/R VQ with Mean Codebook Size 256 and Residual Codebook Size 256 Using Lena Image</td>
<td>124</td>
</tr>
<tr>
<td>4.10</td>
<td>Performance Comparison for M/R VQ with Mean Codebook Size 64 and Residual Codebook Size 64 Using F16, Pepper and Lena Image</td>
<td>125</td>
</tr>
<tr>
<td>4.11</td>
<td>Error for Different Sample Size (Number of Cluster=256)</td>
<td>130</td>
</tr>
<tr>
<td>4.12</td>
<td>Error for Different Number of Clusters (Sample Size = 10000)</td>
<td>131</td>
</tr>
<tr>
<td>4.13</td>
<td>The relationship between the PSNR of the watermarked image and the number of clusters</td>
<td>139</td>
</tr>
<tr>
<td>4.14</td>
<td>NHS Value for Various Attacks</td>
<td>140</td>
</tr>
<tr>
<td>4.15</td>
<td>Comparisons of various fast search algorithms for 'LENA' image in the training set</td>
<td>161</td>
</tr>
<tr>
<td>4.16</td>
<td>Comparisons of various fast search algorithms for 'BABOO' image in the training set</td>
<td>161</td>
</tr>
<tr>
<td>Table</td>
<td>Description</td>
<td>Page</td>
</tr>
<tr>
<td>-------</td>
<td>-------------</td>
<td>------</td>
</tr>
<tr>
<td>5.1</td>
<td>Asymmetric initialization ranges and $V_{\text{max}}$ values</td>
<td>174</td>
</tr>
<tr>
<td>5.2</td>
<td>Performance Comparison of PSO and PPSO with The First Communication Strategy for Rosenbrock Function</td>
<td>175</td>
</tr>
<tr>
<td>5.3</td>
<td>Performance Comparison of PSO and PPSO with The First Communication Strategy for Rastrigin Function</td>
<td>175</td>
</tr>
<tr>
<td>5.4</td>
<td>Performance Comparison of PSO and PPSO with The Second Communication Strategy for Griewank Function</td>
<td>175</td>
</tr>
<tr>
<td>5.5</td>
<td>Performance Comparison of PSO and PPSO with The Third Communication Strategy</td>
<td>176</td>
</tr>
<tr>
<td>6.1</td>
<td>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</td>
<td>192</td>
</tr>
<tr>
<td>6.2</td>
<td>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</td>
<td>193</td>
</tr>
<tr>
<td>6.3</td>
<td>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</td>
<td>194</td>
</tr>
</tbody>
</table>
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 2000a, Estivill-Castro & Lee 2000b, 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 1996c). 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 (Zaâïane & 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-
expected 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 \textit{inter alia} partitioning methods, hierarchical methods, density-based methods, grid-based methods and model-based methods as described as follows (Han & Kamber 2001):

\textbf{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 \textit{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
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 be 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 2000b, 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.
- They 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 discrimi-
nation.
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 Eberhart (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-
CHAPTER 1. INTRODUCTION

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

\[
D_t = \sum_{x_m \in S_p} d(x_m, o_p)
\]

where \(S_p\) is the \(p\)th 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
CHAPTER 2. IMPROVED K-MEDOIDS ALGORITHMS

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:

1. **Step 1: Initialization** - choose k medoids from T objects randomly.
2. **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.
3. **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 $\beta$ 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 $O'(q(k^2s^2 + k(T - k)) + \beta k^2s^2)$ based on the number of distance calculations, where $q$, $s$, $k$, $\beta$ 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 $O((\beta + \text{numlocal})(T-k))$ based on the number of partitions per object or $O'(\beta + \text{numlocal}k(T-k))$ based on the number of distance calculations, where $\beta$ 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

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

$$u_{ij} = \frac{\left(\frac{1}{r(x_j, o_i)}\right)^{\frac{1}{m-1}}}{\sum_{q=1}^{k} \left(\frac{1}{r(x_j, o_q)}\right)^{\frac{1}{m-1}}}, \quad i = 1, \ldots, k, \quad j = 1, \ldots, T,$$

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

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

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

**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

\[ J_m(O, X) = \sum_{j=1}^{T} \left( \sum_{i=1}^{k} \left( r(x_j, o_i) \right)^{-\frac{1}{m}} \right)^{1-m} = \sum_{j=1}^{T} h_j \]  

(2.3)

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

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

\[ J_m^T(O, X) = \sum_{q=1}^{s} h_q \]  

(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

\[ h_j = \left( \sum_{i=1}^{k} \left( r(x_j, o_i) \right)^{-\frac{1}{m}} \right)^{1-m}, \quad j = 1, \ldots, T \]

sort \( h_j \) in the ascending order such that

\[ h_1 \leq h_2 \leq \ldots \leq h_T. \]

**Step 3:** Compute the memberships for \( s \) objects, \( u_{ij} = \frac{\left( \frac{1}{r(x_j, o_i)} \right)^{\frac{1}{m}}}{\sum_{q=1}^{k} \left( \frac{1}{r(x_j, o_q)} \right)^{\frac{1}{m}}}, \]

\[ i = 1, \ldots, k, \quad j = 1, \ldots, s. \]

**Step 4:** Update the new medoids using

\[ p = \arg \min_{1 \leq v \leq s} \sum_{j=1}^{s} u_{vj}^m r(x_v, x_j), \]

\[ o_i = x_p, \quad i = 1, \ldots, k. \]

**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 $\Delta D = D_s(s') - D_s(s)$. 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, \( \eta \) 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 \( 40 + 2 \times k \) 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 \( T_0 \), \( \eta \), 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 $T_0$, $\eta$, 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
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 $T_0$, $\eta$, the number of total distance drop $\text{disdrop, per}$ and the total number of perturbations $\text{totalper}$ 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

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

$q$ 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 $T_0$, $\eta$ 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 $k \ast T$). For CLASA, the parameters for the final temperature $T_f$, the initial temperature $T_0$, $\eta$, 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

<table>
<thead>
<tr>
<th>seed</th>
<th>PAM</th>
<th>CLARA</th>
<th>CLARANS</th>
<th>CLASA</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Average dist.</td>
<td>Count of dist.(10^5)</td>
<td>Average dist.</td>
<td>Count of dist.(10^5)</td>
</tr>
<tr>
<td>1</td>
<td>0.163</td>
<td>1640305</td>
<td>0.194</td>
<td>954</td>
</tr>
<tr>
<td>2</td>
<td>0.164</td>
<td>1333649</td>
<td>0.206</td>
<td>983</td>
</tr>
<tr>
<td>3</td>
<td>0.162</td>
<td>2265256</td>
<td>0.196</td>
<td>940</td>
</tr>
<tr>
<td>4</td>
<td>0.167</td>
<td>1796543</td>
<td>0.200</td>
<td>1084</td>
</tr>
<tr>
<td>5</td>
<td>0.167</td>
<td>1679364</td>
<td>0.217</td>
<td>1084</td>
</tr>
<tr>
<td>6</td>
<td>0.167</td>
<td>1679364</td>
<td>0.205</td>
<td>1069</td>
</tr>
<tr>
<td>7</td>
<td>0.163</td>
<td>2109018</td>
<td>0.196</td>
<td>1069</td>
</tr>
<tr>
<td>8</td>
<td>0.163</td>
<td>1679364</td>
<td>0.219</td>
<td>1041</td>
</tr>
<tr>
<td>9</td>
<td>0.163</td>
<td>1913721</td>
<td>0.205</td>
<td>925</td>
</tr>
<tr>
<td>10</td>
<td>0.164</td>
<td>2148098</td>
<td>0.209</td>
<td>1055</td>
</tr>
<tr>
<td>Ave.</td>
<td>0.164</td>
<td>1824468</td>
<td>0.205</td>
<td>1020</td>
</tr>
</tbody>
</table>

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.
CHAPTER 2. IMPROVED K-MEDOIDS ALGORITHMS

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

<table>
<thead>
<tr>
<th>seed</th>
<th>( PAM )</th>
<th>( CLARA )</th>
<th>( CLARANS )</th>
<th>( CLASA )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Average dist.</td>
<td>Count of dist. ( (10^5) )</td>
<td>Average dist.</td>
<td>Count of dist. ( (10^5) )</td>
</tr>
<tr>
<td>1</td>
<td>4.061</td>
<td>293457</td>
<td>5.053</td>
<td>3355</td>
</tr>
<tr>
<td>2</td>
<td>4.053</td>
<td>326831</td>
<td>5.048</td>
<td>3399</td>
</tr>
<tr>
<td>3</td>
<td>4.066</td>
<td>335175</td>
<td>5.261</td>
<td>3532</td>
</tr>
<tr>
<td>4</td>
<td>4.038</td>
<td>318488</td>
<td>4.971</td>
<td>3266</td>
</tr>
<tr>
<td>5</td>
<td>4.045</td>
<td>343519</td>
<td>5.055</td>
<td>4153</td>
</tr>
<tr>
<td>6</td>
<td>4.027</td>
<td>326831</td>
<td>4.988</td>
<td>3710</td>
</tr>
<tr>
<td>7</td>
<td>4.036</td>
<td>293457</td>
<td>4.930</td>
<td>3576</td>
</tr>
<tr>
<td>8</td>
<td>4.050</td>
<td>385236</td>
<td>5.065</td>
<td>3665</td>
</tr>
<tr>
<td>9</td>
<td>4.037</td>
<td>368549</td>
<td>5.215</td>
<td>4020</td>
</tr>
<tr>
<td>10</td>
<td>4.042</td>
<td>376893</td>
<td>5.146</td>
<td>4065</td>
</tr>
<tr>
<td>Ave.</td>
<td>4.046</td>
<td>336844</td>
<td>5.073</td>
<td>3674</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>seed</th>
<th>CLARA</th>
<th></th>
<th>CLARANS</th>
<th></th>
<th>CLASA</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Average dist.</td>
<td>Count of dist.(10^5)</td>
<td>Average dist.</td>
<td>Count of dist.(10^5)</td>
<td>Average dist.</td>
<td>Count of dist.(10^5)</td>
</tr>
<tr>
<td>1</td>
<td>0.228</td>
<td>2004</td>
<td>0.228</td>
<td>1609</td>
<td>0.221</td>
<td>971</td>
</tr>
<tr>
<td>2</td>
<td>0.221</td>
<td>2302</td>
<td>0.232</td>
<td>1067</td>
<td>0.224</td>
<td>991</td>
</tr>
<tr>
<td>3</td>
<td>0.236</td>
<td>2388</td>
<td>0.236</td>
<td>1351</td>
<td>0.235</td>
<td>965</td>
</tr>
<tr>
<td>4</td>
<td>0.230</td>
<td>2686</td>
<td>0.233</td>
<td>1154</td>
<td>0.222</td>
<td>976</td>
</tr>
<tr>
<td>5</td>
<td>0.227</td>
<td>2430</td>
<td>0.232</td>
<td>1572</td>
<td>0.222</td>
<td>993</td>
</tr>
<tr>
<td>6</td>
<td>0.237</td>
<td>2260</td>
<td>0.227</td>
<td>1223</td>
<td>0.224</td>
<td>947</td>
</tr>
<tr>
<td>7</td>
<td>0.232</td>
<td>2464</td>
<td>0.234</td>
<td>895</td>
<td>0.222</td>
<td>973</td>
</tr>
<tr>
<td>8</td>
<td>0.236</td>
<td>2516</td>
<td>0.231</td>
<td>1429</td>
<td>0.231</td>
<td>927</td>
</tr>
<tr>
<td>9</td>
<td>0.233</td>
<td>2345</td>
<td>0.231</td>
<td>1148</td>
<td>0.223</td>
<td>993</td>
</tr>
<tr>
<td>10</td>
<td>0.238</td>
<td>2728</td>
<td>0.231</td>
<td>1167</td>
<td>0.227</td>
<td>994</td>
</tr>
<tr>
<td>Ave.</td>
<td>0.232</td>
<td>2431</td>
<td>0.232</td>
<td>1140</td>
<td>0.225</td>
<td>973</td>
</tr>
</tbody>
</table>

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

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

<table>
<thead>
<tr>
<th>Order of Closeness to ( x_j )</th>
<th>Result of Swap</th>
<th>Cost</th>
<th>Case</th>
</tr>
</thead>
<tbody>
<tr>
<td>( o_{old} \times o_{new} \times o_{m} )</td>
<td>Cluster representation transfers from ( o_{old} ) to ( o_{new} )</td>
<td>( d(x_j, o_{new}) - d(x_j, o_{old}) )</td>
<td>2</td>
</tr>
<tr>
<td>( o_{new} \times o_{old} \times o_{m} )</td>
<td>Cluster representation transfers from ( o_{old} ) to ( o_{new} )</td>
<td>( d(x_j, o_{new}) - d(x_j, o_{old}) )</td>
<td>2</td>
</tr>
<tr>
<td>( o_{old} \times o_{m} \times o_{new} )</td>
<td>Cluster representation transfers from ( o_{old} ) to ( o_{m} )</td>
<td>( d(x_j, o_{m}) - d(x_j, o_{old}) )</td>
<td>1</td>
</tr>
<tr>
<td>( o_{new} \times o_{m} \times o_{old} )</td>
<td>Cluster representation transfers from ( o_{m} ) to ( o_{new} )</td>
<td>( d(x_j, o_{new}) - d(x_j, o_{old}) )</td>
<td>4</td>
</tr>
<tr>
<td>( o_{m} \times { o_{new} \times o_{old} } )</td>
<td>Cluster representation remains with ( o_{m} )</td>
<td>nil</td>
<td>3</td>
</tr>
</tbody>
</table>
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

$$C_{old} = d(o_{old}, o_m) - d(o_{old}, o_{old}) = d(o_{old}, o_m)$$  \hspace{1cm} (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

$$C_{old} = d(o_{old}, o_{new}) - d(o_{old}, o_{old}) = d(o_{new}, o_{old})$$  \hspace{1cm} (2.6)

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

$$D'_t - D_t = C_{old} + \sum_j C_j$$  \hspace{1cm} (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 1996b, 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:

$$D(X, C_t) = \sum_{i=1}^{k} (x^i - c^i_t)^2,$$

(2.8)

where $C_t = \{c^1_t, c^2_t, ..., c^k_t\}$ 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. 1996c, Pan et al. 1996b, Pan, Lu & Sun 2003, Wu & Lin 2000, Lu & Sun 2003), pyramid structure based (Lee & Chen 1995, Pan, Lu & Sun 2000, Song & Ra 2002a) and transform domain inequality based (Hwang, Jeng & Chen 1997, Lu, Pan & Sun 2000a, 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. 1996b), 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

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

(2.9)

if

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

(2.10)

then

$$D(x, o_j) \geq D(x, o_t),$$

(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:

\begin{align}
    d(x, y) &\geq 0; d(x, y) = 0 \text{ if } 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}
\end{align}

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

  \[ d(o_1, o_2) \leq d(t, o_2) + d(t, o_1) \tag{2.15} \]
if
\[ d(o_1, o_2) \geq 2d(t, o_1) \]  \hspace{1cm} (2.16)
then
\[ d(t, o_2) \geq d(t, o_1) \]  \hspace{1cm} (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 2002b), 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
\[ d^2(o_1, o_2)/4 \geq d^2(x, o_1) \]  \hspace{1cm} (2.18)
then
\[ d(x, o_2) \geq d(x, o_1) \]  \hspace{1cm} (2.19)
CHAPTER 2. IMPROVED K-MEDOIDS ALGORITHMS

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

Given

$$d(o_3, o_2) \geq d(t, o_2) - d(t, o_3); \quad (2.23)$$

if

$$d(o_3, o_2) \leq d(t, o_2) - d(t, o_1) \quad (2.24)$$

then

$$d(t, o_3) \geq d(t, o_1) \quad (2.25)$$

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. 1996c). By merging criteria 2 and 3, we may get the following criterion, i.e.,
CHAPTER 2. IMPROVED K-MEDOIDS ALGORITHMS

\[ \text{if \ } d(x, o_1) \leq \left| d(o_3, o_2) - d(x, o_2) \right| \]  \quad (2.26)

\[ \text{then \ } d(x, o_3) \geq d(x, o_1) \]  \quad (2.27)

Set

\[ o_2 = \vec{0} \quad \text{(zero vector).} \]

Hence

\[ \text{if \ } d(x, o_1) \leq \left| d(o_3, \vec{o}) - d(x, \vec{o}) \right|, \]  \quad (2.28)

\[ \text{then \ } d(x, o_3) \geq d(x, o_1). \]  \quad (2.29)

For Euclidean distance measure and given

\[ d_{\text{min}} = d(x, o_1), \]

\[ \text{if \ } d_{\text{min}} \leq \left| \sqrt{\sum_{i=1}^{d} o_{3i}^2} - \sqrt{\sum_{i=1}^{d} x_i^2} \right|, \]  \quad (2.30)

\[ \text{then \ } d(x, o_3) \geq d_{\text{min}}. \]  \quad (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 criteria 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
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-
CHAPTER 2. IMPROVED K-MEDOIDS ALGORITHMS

Figure 2.16. Usage of Memory for Distance Calculations

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

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

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

$$D_{t1}' = \sum_{i=1}^{T-k} \min(d(x_i, NM(x_i)), d(x_i, o_{new})). \quad (2.33)$$
CHAPTER 2. IMPROVED K-MEDOIDS ALGORITHMS

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

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

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

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

$$D'_t = \sum_{i=1}^{T-k} \left\{ \min(d(x_i, NM(x_i)), d(x_i, o_{new})) + d(x_i, o_p) \right\} + \min(d(o_{old}, o_j)), j = 1, \ldots k$$  \hfill (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 \((kn \times 1.25\%), \text{ where } n \text{ is the number of objects} \), respectively. As shown in Table 2.6, CLARANS-MITP, CLARANS-M and CLARANS-ITP performed better than CLARA both in the distance calculations and the average distance per object. Comparing with CLARANS, CLARANS-MITP, CLARANS-M and CLARANS-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-MITP, CLARANS-M and CLARANS-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 \(T_0\), \(\eta\), 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 $T_0$, $\eta$, 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
CHAPTER 2. IMPROVED K-MEDOIDS ALGORITHMS

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.
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-MITP, CLARANS-M and CLARANS-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 understanding 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-MITP, CLARANS-M and CLARANS-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-MITP, CLARANS-M and CLARANS-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-MITP, CLARANS-M and CLARANS-ITP outperformed better than CLARANS with the same average distance per object. Comparing with CLARANS, CLARANS-MITP, CLARANS-M and CLARANS-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 \textit{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 \textit{CLARANS} can reduce the number of distance computations from 87\% to 98\%. In terms of the running time, the proposed \textit{CLARANS-MITP} algorithm can reduce the computation time up to 96\%. Experiments with \textit{CLASA} also indicate an improvement over \textit{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 \textit{CLARA} and \textit{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

<table>
<thead>
<tr>
<th>Seed</th>
<th>CLARA</th>
<th>CLARANS</th>
<th>CLARANS-ITP</th>
<th>CLARANS-M</th>
<th>CLARANS-MITP</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Distance Calcs ($10^5$)</td>
<td>Distance Calcs ($10^5$)</td>
<td>Distance Calcs ($10^5$)</td>
<td>Distance Calcs ($10^5$)</td>
<td>Distance Calcs ($10^5$)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Average Distance</td>
<td>Running Time (secs)</td>
<td>Running Time (secs)</td>
<td>Running Time (secs)</td>
<td>Running Time (secs)</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>851</td>
<td>4.748</td>
<td>16</td>
<td>205</td>
<td>4</td>
<td>99</td>
</tr>
<tr>
<td>2</td>
<td>779</td>
<td>4.720</td>
<td>15</td>
<td>247</td>
<td>5</td>
<td>119</td>
</tr>
<tr>
<td>3</td>
<td>880</td>
<td>4.595</td>
<td>17</td>
<td>201</td>
<td>4</td>
<td>98</td>
</tr>
<tr>
<td>4</td>
<td>807</td>
<td>4.698</td>
<td>15</td>
<td>200</td>
<td>4</td>
<td>97</td>
</tr>
<tr>
<td>5</td>
<td>721</td>
<td>4.714</td>
<td>14</td>
<td>249</td>
<td>5</td>
<td>121</td>
</tr>
<tr>
<td>6</td>
<td>836</td>
<td>4.753</td>
<td>16</td>
<td>194</td>
<td>4</td>
<td>93</td>
</tr>
<tr>
<td>7</td>
<td>908</td>
<td>4.829</td>
<td>17</td>
<td>289</td>
<td>6</td>
<td>141</td>
</tr>
<tr>
<td>8</td>
<td>908</td>
<td>4.781</td>
<td>17</td>
<td>231</td>
<td>5</td>
<td>110</td>
</tr>
<tr>
<td>9</td>
<td>836</td>
<td>4.788</td>
<td>16</td>
<td>295</td>
<td>6</td>
<td>143</td>
</tr>
<tr>
<td>10</td>
<td>880</td>
<td>4.732</td>
<td>17</td>
<td>265</td>
<td>5</td>
<td>128</td>
</tr>
<tr>
<td>Ave.</td>
<td>841</td>
<td>4.736</td>
<td>16</td>
<td>238</td>
<td>5</td>
<td>115</td>
</tr>
</tbody>
</table>
### Table 2.7. Results of Experiment for Gauss-Markov Source

| Seed | Distance Calcs ($10^7$) | Average Distance | Running Time (secs) | Distance Calcs ($10^7$) | Running Distance | Running Time (secs) | Distance Calcs ($10^7$) | Running Distance | Running Time (secs) | Distance Calcs ($10^7$) | Running Distance | Running Time (secs) | Distance Calcs ($10^7$) | Running Distance | Running Time (secs) | Average Distance |
|------|-------------------------|------------------|--------------------|-------------------------|------------------------|--------------------|-------------------------|------------------------|--------------------|-------------------------|------------------------|--------------------|--------------------|-------------------------|------------------------|--------------------|------------------|
| 1    | 1548                    | 4.559            | 2790               | 376                    | 662                    | 126                | 411                     | 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

<table>
<thead>
<tr>
<th>Seed</th>
<th>CLARANS</th>
<th>CLASA</th>
<th>CLASA-ITP</th>
<th>CLASA-M</th>
<th>CLASA-MITP</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Distance Calcs (10^7)</td>
<td>Distance Running Time (secs)</td>
<td>Distance Running Time (secs)</td>
<td>Distance Running Time (secs)</td>
<td>Distance Running Time (secs)</td>
<td>Average Distance</td>
</tr>
<tr>
<td>1</td>
<td>376</td>
<td>4.332</td>
<td>662</td>
<td>243</td>
<td>479</td>
<td>15</td>
</tr>
<tr>
<td>2</td>
<td>532</td>
<td>4.359</td>
<td>934</td>
<td>242</td>
<td>479</td>
<td>15</td>
</tr>
<tr>
<td>3</td>
<td>375</td>
<td>4.381</td>
<td>639</td>
<td>241</td>
<td>475</td>
<td>15</td>
</tr>
<tr>
<td>4</td>
<td>406</td>
<td>4.398</td>
<td>713</td>
<td>242</td>
<td>479</td>
<td>15</td>
</tr>
<tr>
<td>5</td>
<td>397</td>
<td>4.367</td>
<td>697</td>
<td>242</td>
<td>477</td>
<td>15</td>
</tr>
<tr>
<td>6</td>
<td>414</td>
<td>4.384</td>
<td>727</td>
<td>241</td>
<td>478</td>
<td>15</td>
</tr>
<tr>
<td>7</td>
<td>323</td>
<td>4.380</td>
<td>568</td>
<td>245</td>
<td>483</td>
<td>15</td>
</tr>
<tr>
<td>8</td>
<td>396</td>
<td>4.394</td>
<td>697</td>
<td>241</td>
<td>478</td>
<td>15</td>
</tr>
<tr>
<td>9</td>
<td>367</td>
<td>4.377</td>
<td>646</td>
<td>242</td>
<td>478</td>
<td>15</td>
</tr>
<tr>
<td>10</td>
<td>358</td>
<td>4.406</td>
<td>634</td>
<td>242</td>
<td>478</td>
<td>15</td>
</tr>
<tr>
<td>Ave.</td>
<td>394</td>
<td>4.388</td>
<td>694</td>
<td>242</td>
<td>478</td>
<td>15</td>
</tr>
</tbody>
</table>
Table 2.9. Results of Experiment for Rectangular Clusters

<table>
<thead>
<tr>
<th>Seed</th>
<th>CLARA</th>
<th>CLARANS</th>
<th>CLARANS-ITP</th>
<th>CLARANS-M</th>
<th>CLARANS-MITP</th>
<th>Average Distance</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Distance</td>
<td>Average</td>
<td>Running</td>
<td>Distance</td>
<td>Running</td>
<td>Distance</td>
</tr>
<tr>
<td></td>
<td>Calcs (10^7)</td>
<td>Time (secs)</td>
<td>Calcs (10^7)</td>
<td>Time (secs)</td>
<td>Calcs (10^7)</td>
<td>Time (secs)</td>
</tr>
<tr>
<td>1</td>
<td>192</td>
<td>0.181</td>
<td>110</td>
<td>167</td>
<td>97</td>
<td>18</td>
</tr>
<tr>
<td>2</td>
<td>205</td>
<td>0.185</td>
<td>120</td>
<td>132</td>
<td>76</td>
<td>14</td>
</tr>
<tr>
<td>3</td>
<td>169</td>
<td>0.182</td>
<td>97</td>
<td>120</td>
<td>70</td>
<td>13</td>
</tr>
<tr>
<td>4</td>
<td>219</td>
<td>0.186</td>
<td>127</td>
<td>138</td>
<td>80</td>
<td>15</td>
</tr>
<tr>
<td>5</td>
<td>174</td>
<td>0.184</td>
<td>101</td>
<td>178</td>
<td>103</td>
<td>19</td>
</tr>
<tr>
<td>6</td>
<td>205</td>
<td>0.181</td>
<td>118</td>
<td>144</td>
<td>84</td>
<td>15</td>
</tr>
<tr>
<td>7</td>
<td>172</td>
<td>0.183</td>
<td>100</td>
<td>155</td>
<td>90</td>
<td>17</td>
</tr>
<tr>
<td>8</td>
<td>205</td>
<td>0.183</td>
<td>119</td>
<td>157</td>
<td>91</td>
<td>17</td>
</tr>
<tr>
<td>9</td>
<td>207</td>
<td>0.185</td>
<td>120</td>
<td>147</td>
<td>85</td>
<td>16</td>
</tr>
<tr>
<td>10</td>
<td>180</td>
<td>0.182</td>
<td>104</td>
<td>177</td>
<td>103</td>
<td>19</td>
</tr>
<tr>
<td>Ave.</td>
<td>193</td>
<td>0.183</td>
<td>112</td>
<td>151</td>
<td>88</td>
<td>16</td>
</tr>
</tbody>
</table>
Table 2.10: Results of Experiment for Rectangular Clusters—Compared with CLARANS and Extended CLASA Algorithms

<table>
<thead>
<tr>
<th>Seed</th>
<th>CLARANS</th>
<th>CLASA</th>
<th>CLASA-ITP</th>
<th>CLASA-M</th>
<th>CLASA-MITP</th>
<th>Average Distance</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Distance Calcs ($10^7$)</td>
<td>Average Distance</td>
<td>Running Time (secs)</td>
<td>Distance Calcs ($10^7$)</td>
<td>Running Time (secs)</td>
<td>Distance Calcs ($10^7$)</td>
</tr>
<tr>
<td>1</td>
<td>167</td>
<td>0.172</td>
<td>97</td>
<td>134</td>
<td>88</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>132</td>
<td>0.177</td>
<td>76</td>
<td>136</td>
<td>88</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>120</td>
<td>0.169</td>
<td>70</td>
<td>136</td>
<td>89</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>138</td>
<td>0.173</td>
<td>80</td>
<td>134</td>
<td>87</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>178</td>
<td>0.172</td>
<td>103</td>
<td>132</td>
<td>86</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>144</td>
<td>0.174</td>
<td>84</td>
<td>133</td>
<td>87</td>
<td>2</td>
</tr>
<tr>
<td>7</td>
<td>155</td>
<td>0.176</td>
<td>90</td>
<td>132</td>
<td>86</td>
<td>2</td>
</tr>
<tr>
<td>8</td>
<td>157</td>
<td>0.175</td>
<td>91</td>
<td>136</td>
<td>89</td>
<td>2</td>
</tr>
<tr>
<td>9</td>
<td>147</td>
<td>0.176</td>
<td>85</td>
<td>135</td>
<td>88</td>
<td>2</td>
</tr>
<tr>
<td>10</td>
<td>177</td>
<td>0.175</td>
<td>103</td>
<td>136</td>
<td>88</td>
<td>2</td>
</tr>
<tr>
<td>Ave.</td>
<td>151</td>
<td>0.174</td>
<td>88</td>
<td>134</td>
<td>87</td>
<td>2</td>
</tr>
</tbody>
</table>
Table 2.11. Results of Experiment for Elliptic Clusters

<table>
<thead>
<tr>
<th>Seed</th>
<th>CLARA</th>
<th>CLARANS</th>
<th>CLARANS-ITP</th>
<th>CLARANS-M</th>
<th>CLARANS-MITP</th>
<th>Average Distance</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Distance Calcs ($10^7$)</td>
<td>Average Distance</td>
<td>Running Time (secs)</td>
<td>Distance Calcs ($10^7$)</td>
<td>Running Time (secs)</td>
<td>Distance Calcs ($10^7$)</td>
</tr>
<tr>
<td>1</td>
<td>1157</td>
<td>0.940</td>
<td>684</td>
<td>918</td>
<td>715</td>
<td>143</td>
</tr>
<tr>
<td>2</td>
<td>925</td>
<td>0.942</td>
<td>547</td>
<td>635</td>
<td>483</td>
<td>100</td>
</tr>
<tr>
<td>3</td>
<td>1225</td>
<td>0.956</td>
<td>721</td>
<td>769</td>
<td>584</td>
<td>120</td>
</tr>
<tr>
<td>4</td>
<td>1034</td>
<td>0.951</td>
<td>612</td>
<td>796</td>
<td>603</td>
<td>125</td>
</tr>
<tr>
<td>5</td>
<td>1116</td>
<td>0.946</td>
<td>659</td>
<td>900</td>
<td>686</td>
<td>142</td>
</tr>
<tr>
<td>6</td>
<td>1089</td>
<td>0.960</td>
<td>647</td>
<td>721</td>
<td>548</td>
<td>114</td>
</tr>
<tr>
<td>7</td>
<td>776</td>
<td>0.972</td>
<td>455</td>
<td>849</td>
<td>635</td>
<td>132</td>
</tr>
<tr>
<td>8</td>
<td>939</td>
<td>0.944</td>
<td>556</td>
<td>931</td>
<td>714</td>
<td>146</td>
</tr>
<tr>
<td>9</td>
<td>844</td>
<td>0.954</td>
<td>493</td>
<td>592</td>
<td>449</td>
<td>94</td>
</tr>
<tr>
<td>10</td>
<td>844</td>
<td>0.942</td>
<td>489</td>
<td>751</td>
<td>576</td>
<td>118</td>
</tr>
<tr>
<td>Ave.</td>
<td>995</td>
<td>0.951</td>
<td>586</td>
<td>786</td>
<td>599</td>
<td>123</td>
</tr>
</tbody>
</table>
Table 2.12. Results of Experiment for Curved Clusters

<table>
<thead>
<tr>
<th>Seed</th>
<th>CLARA</th>
<th>CLARANS</th>
<th>CLARANS-ITP</th>
<th>CLARANS-M</th>
<th>CLARANS-MITP</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Distance Calcs ($10^7$)</td>
<td>Average Distance</td>
<td>Running Time (secs)</td>
<td>Distance Calcs ($10^7$)</td>
<td>Running Time (secs)</td>
<td>Distance Calcs ($10^7$)</td>
</tr>
<tr>
<td>1</td>
<td>741</td>
<td>2.232</td>
<td>425</td>
<td>652</td>
<td>397</td>
<td>53</td>
</tr>
<tr>
<td>2</td>
<td>734</td>
<td>2.278</td>
<td>432</td>
<td>626</td>
<td>372</td>
<td>51</td>
</tr>
<tr>
<td>3</td>
<td>847</td>
<td>2.317</td>
<td>498</td>
<td>702</td>
<td>422</td>
<td>57</td>
</tr>
<tr>
<td>4</td>
<td>797</td>
<td>2.342</td>
<td>463</td>
<td>719</td>
<td>425</td>
<td>58</td>
</tr>
<tr>
<td>5</td>
<td>840</td>
<td>2.321</td>
<td>475</td>
<td>574</td>
<td>341</td>
<td>45</td>
</tr>
<tr>
<td>6</td>
<td>805</td>
<td>2.252</td>
<td>465</td>
<td>623</td>
<td>368</td>
<td>49</td>
</tr>
<tr>
<td>7</td>
<td>741</td>
<td>2.305</td>
<td>432</td>
<td>650</td>
<td>398</td>
<td>52</td>
</tr>
<tr>
<td>8</td>
<td>797</td>
<td>2.238</td>
<td>463</td>
<td>636</td>
<td>378</td>
<td>51</td>
</tr>
<tr>
<td>9</td>
<td>741</td>
<td>2.269</td>
<td>429</td>
<td>551</td>
<td>339</td>
<td>44</td>
</tr>
<tr>
<td>10</td>
<td>805</td>
<td>2.317</td>
<td>473</td>
<td>484</td>
<td>285</td>
<td>38</td>
</tr>
<tr>
<td>Ave.</td>
<td>785</td>
<td>2.281</td>
<td>455</td>
<td>622</td>
<td>373</td>
<td>50</td>
</tr>
</tbody>
</table>
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 Distance Calcs ($10^7$) | CLARA Average Distance | CLARA Running Time (secs) | CLARANS Distance Calcs ($10^7$) | CLARANS Running Time (secs) | CLARANS-ITP Distance Calcs ($10^7$) | CLARANS-ITP Running Time (secs) | CLARANS-M Distance Calcs ($10^7$) | CLARANS-M Running Time (secs) | CLARANS-MITP Distance Calcs ($10^7$) | CLARANS-MITP Running Time (secs) | Average Distance
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>299</td>
<td>2829.559</td>
<td>1014</td>
<td>797</td>
<td>2985</td>
<td>183</td>
<td>1197</td>
<td>187</td>
<td>1108</td>
<td>130</td>
<td>949</td>
<td>2816.426</td>
</tr>
<tr>
<td>2</td>
<td>312</td>
<td>2833.360</td>
<td>1057</td>
<td>568</td>
<td>2127</td>
<td>130</td>
<td>853</td>
<td>133</td>
<td>795</td>
<td>92</td>
<td>678</td>
<td>2831.768</td>
</tr>
<tr>
<td>3</td>
<td>325</td>
<td>2839.221</td>
<td>1100</td>
<td>761</td>
<td>2853</td>
<td>174</td>
<td>1133</td>
<td>179</td>
<td>1066</td>
<td>123</td>
<td>904</td>
<td>2806.479</td>
</tr>
<tr>
<td>4</td>
<td>358</td>
<td>2821.522</td>
<td>1216</td>
<td>597</td>
<td>2236</td>
<td>137</td>
<td>896</td>
<td>140</td>
<td>835</td>
<td>97</td>
<td>710</td>
<td>2814.164</td>
</tr>
<tr>
<td>5</td>
<td>286</td>
<td>2827.433</td>
<td>969</td>
<td>442</td>
<td>1656</td>
<td>102</td>
<td>664</td>
<td>104</td>
<td>619</td>
<td>72</td>
<td>530</td>
<td>2821.410</td>
</tr>
<tr>
<td>6</td>
<td>286</td>
<td>2842.224</td>
<td>972</td>
<td>544</td>
<td>2034</td>
<td>123</td>
<td>809</td>
<td>127</td>
<td>750</td>
<td>87</td>
<td>643</td>
<td>2822.102</td>
</tr>
<tr>
<td>7</td>
<td>306</td>
<td>2838.232</td>
<td>1257</td>
<td>559</td>
<td>2094</td>
<td>128</td>
<td>835</td>
<td>131</td>
<td>776</td>
<td>90</td>
<td>666</td>
<td>2817.135</td>
</tr>
<tr>
<td>8</td>
<td>286</td>
<td>2843.913</td>
<td>1179</td>
<td>568</td>
<td>2128</td>
<td>131</td>
<td>856</td>
<td>133</td>
<td>789</td>
<td>93</td>
<td>680</td>
<td>2819.383</td>
</tr>
<tr>
<td>9</td>
<td>247</td>
<td>2820.759</td>
<td>1210</td>
<td>646</td>
<td>2419</td>
<td>148</td>
<td>968</td>
<td>151</td>
<td>905</td>
<td>104</td>
<td>769</td>
<td>2823.581</td>
</tr>
<tr>
<td>10</td>
<td>286</td>
<td>2819.524</td>
<td>966</td>
<td>590</td>
<td>2210</td>
<td>135</td>
<td>884</td>
<td>138</td>
<td>819</td>
<td>96</td>
<td>701</td>
<td>2826.953</td>
</tr>
<tr>
<td>Ave.</td>
<td>299</td>
<td>2831.575</td>
<td>1094</td>
<td>607</td>
<td>2274</td>
<td>139</td>
<td>910</td>
<td>142</td>
<td>846</td>
<td>98</td>
<td>723</td>
<td>2819.940</td>
</tr>
</tbody>
</table>
Table 2.14: Experimental Results for Ten Runs of CLARANS, CLARANS-ITP, CLARANS-M and CLARANS-MITP Algorithms for 16 Medoids of Lena Dataset

<table>
<thead>
<tr>
<th>Seed</th>
<th>CLARA</th>
<th>CLARANS</th>
<th>CLARANS-ITP</th>
<th>CLARANS-M</th>
<th>CLARANS-MITP</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Distance</td>
<td>Average</td>
<td>Running</td>
<td>Distance</td>
<td>Average</td>
</tr>
<tr>
<td></td>
<td>Calcs ($10^7$)</td>
<td>Distance Calcs ($10^7$)</td>
<td>Time (secs)</td>
<td>Distance Calcs ($10^7$)</td>
<td>Time (secs)</td>
</tr>
<tr>
<td>1</td>
<td>2220</td>
<td>2108.098</td>
<td>8328</td>
<td>1249</td>
<td>4984</td>
</tr>
<tr>
<td>2</td>
<td>2273</td>
<td>2093.875</td>
<td>8541</td>
<td>1530</td>
<td>6110</td>
</tr>
<tr>
<td>3</td>
<td>2114</td>
<td>2115.623</td>
<td>7943</td>
<td>1417</td>
<td>5651</td>
</tr>
<tr>
<td>4</td>
<td>2379</td>
<td>2070.973</td>
<td>7936</td>
<td>1216</td>
<td>4850</td>
</tr>
<tr>
<td>5</td>
<td>2114</td>
<td>2158.374</td>
<td>7033</td>
<td>1756</td>
<td>7010</td>
</tr>
<tr>
<td>6</td>
<td>2643</td>
<td>2071.464</td>
<td>8791</td>
<td>1313</td>
<td>5204</td>
</tr>
<tr>
<td>7</td>
<td>2167</td>
<td>2140.907</td>
<td>7224</td>
<td>1736</td>
<td>6884</td>
</tr>
<tr>
<td>8</td>
<td>2009</td>
<td>2097.309</td>
<td>6742</td>
<td>1477</td>
<td>5683</td>
</tr>
<tr>
<td>9</td>
<td>1982</td>
<td>2126.558</td>
<td>6633</td>
<td>2018</td>
<td>8009</td>
</tr>
<tr>
<td>10</td>
<td>1824</td>
<td>2089.289</td>
<td>6832</td>
<td>1312</td>
<td>4632</td>
</tr>
<tr>
<td>Ave.</td>
<td>2172</td>
<td>2107.247</td>
<td>7600</td>
<td>1502</td>
<td>5920</td>
</tr>
</tbody>
</table>

Table 2.15: Results of Experiment for Lena Dataset

<table>
<thead>
<tr>
<th>Medoids</th>
<th>CLARANS</th>
<th>CLARANS-ITP</th>
<th>CLARANS-M</th>
<th>CLARANS-MITP</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Distance Calcs ($10^7$)</td>
<td>Running Time (secs)</td>
<td>Distance Calcs ($10^7$)</td>
<td>Running Time (secs)</td>
</tr>
<tr>
<td>64</td>
<td>10577</td>
<td>35458</td>
<td>530</td>
<td>4480</td>
</tr>
<tr>
<td>32</td>
<td>4133</td>
<td>14181</td>
<td>326</td>
<td>2482</td>
</tr>
<tr>
<td>16</td>
<td>1502</td>
<td>5920</td>
<td>196</td>
<td>1385</td>
</tr>
<tr>
<td>8</td>
<td>607</td>
<td>2274</td>
<td>139</td>
<td>910</td>
</tr>
</tbody>
</table>
Table 2.16. Results of Experiment for Co-Occurrence Texture of Corel Image Collection

<table>
<thead>
<tr>
<th>Seed</th>
<th>CLARANS</th>
<th>CLARANS-ITP</th>
<th>CLARANS-M</th>
<th>CLARANS-MITP</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Distance Calcs ($10^7$)</td>
<td>Running Time (secs)</td>
<td>Distance Calcs ($10^7$)</td>
<td>Running Time (secs)</td>
</tr>
<tr>
<td>1</td>
<td>12557</td>
<td>43754</td>
<td>2793</td>
<td>16473</td>
</tr>
<tr>
<td>2</td>
<td>18523</td>
<td>64562</td>
<td>4072</td>
<td>23900</td>
</tr>
<tr>
<td>3</td>
<td>13845</td>
<td>48110</td>
<td>3088</td>
<td>18187</td>
</tr>
<tr>
<td>4</td>
<td>19136</td>
<td>66388</td>
<td>4287</td>
<td>25336</td>
</tr>
<tr>
<td>5</td>
<td>15938</td>
<td>55768</td>
<td>3563</td>
<td>20953</td>
</tr>
<tr>
<td>6</td>
<td>15844</td>
<td>55024</td>
<td>3519</td>
<td>20498</td>
</tr>
<tr>
<td>7</td>
<td>15097</td>
<td>52272</td>
<td>3391</td>
<td>26678</td>
</tr>
<tr>
<td>8</td>
<td>11963</td>
<td>41403</td>
<td>2640</td>
<td>20762</td>
</tr>
<tr>
<td>9</td>
<td>13528</td>
<td>46781</td>
<td>2981</td>
<td>17519</td>
</tr>
<tr>
<td>10</td>
<td>18400</td>
<td>63728</td>
<td>4070</td>
<td>23739</td>
</tr>
<tr>
<td>Ave.</td>
<td>15483</td>
<td>53779</td>
<td>3440</td>
<td>21404</td>
</tr>
</tbody>
</table>
CHAPTER 2. IMPROVED K-MEDOIDS ALGORITHMS

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 areas. 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 center 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 2002a, Chu, Roddick & Pan 2004c) 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 $\text{NumCandidate}$ 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 $\text{NumSample}$ 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 \textit{NumRun} times.

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

Step 3: Get \textit{NumCandidate} objects for each cluster by selecting the \textit{NumCandidate} nearest objects from the centroid in each cluster.

Step 4: Repeat the following steps \textit{NumSample} times.

Step 5: Generate medoids by selecting one object from the \textit{NumCandidate} nearest objects in each cluster.

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

\textit{MCMRS} sampling scheme can be combined with \textit{PAM}, \textit{CLARA}, \textit{CLARANS} and \textit{CLASA} to further improve the performance. The combined version of \textit{MCMRS} and \textit{CLASA} for example, (referred here as \textit{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 \textit{NumRun} times.

Step 2: Get \textit{NumCandidate} objects for each cluster by selecting the \textit{NumCandidate} nearest objects from the centroid in each cluster.

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

Step 4: Terminate the program and return the medoids when the predefined criterion is satisfied.
CHAPTER 3. SAMPLING SCHEMES FOR K-MEDOIDS ALGORITHM

3.2.2 Experimental Results

Three artificial databases and one real image database were used for the experiments 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. $1.5\% \times k \times T$). For CLASA, the parameters for the final temperature $T_f$, the initial temperature $T_0$, $\eta$, the times of distance drops $\text{dis\_drop}$,
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>
<thead>
<tr>
<th>seed</th>
<th>CLARA</th>
<th>CLARANS</th>
<th>MCMRS</th>
<th>MCMRS-CLASA</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Average dist.</td>
<td>Count of dist.(10^5)</td>
<td>Average dist.</td>
<td>Count of dist.(10^5)</td>
</tr>
<tr>
<td>1</td>
<td>0.228</td>
<td>2004</td>
<td>0.228</td>
<td>1609</td>
</tr>
<tr>
<td>2</td>
<td>0.221</td>
<td>2302</td>
<td>0.232</td>
<td>1067</td>
</tr>
<tr>
<td>3</td>
<td>0.236</td>
<td>2388</td>
<td>0.236</td>
<td>1351</td>
</tr>
<tr>
<td>4</td>
<td>0.230</td>
<td>2686</td>
<td>0.233</td>
<td>1154</td>
</tr>
<tr>
<td>5</td>
<td>0.227</td>
<td>2430</td>
<td>0.232</td>
<td>1572</td>
</tr>
<tr>
<td>6</td>
<td>0.237</td>
<td>2260</td>
<td>0.227</td>
<td>1223</td>
</tr>
<tr>
<td>7</td>
<td>0.232</td>
<td>2646</td>
<td>0.234</td>
<td>895</td>
</tr>
<tr>
<td>8</td>
<td>0.236</td>
<td>2516</td>
<td>0.231</td>
<td>1429</td>
</tr>
<tr>
<td>9</td>
<td>0.233</td>
<td>2345</td>
<td>0.231</td>
<td>1148</td>
</tr>
<tr>
<td>10</td>
<td>0.238</td>
<td>2728</td>
<td>0.231</td>
<td>1167</td>
</tr>
<tr>
<td>Ave.</td>
<td>0.232</td>
<td>2431</td>
<td>0.232</td>
<td>1140</td>
</tr>
</tbody>
</table>

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-
CHAPTER 3. SAMPLING SCHEMES FOR K-MEDOIDS ALGORITHM

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

respectively. 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 $T_0$, $\eta$, 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 $200 + 2 \times k$. 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

<table>
<thead>
<tr>
<th>seed</th>
<th>CLARA</th>
<th>CLARANS</th>
<th>MCMRS</th>
<th>MCMRS-CLASA</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Average dist.</td>
<td>Average dist.</td>
<td>Average dist.</td>
<td>Average dist.</td>
</tr>
<tr>
<td></td>
<td>Count of dist.(10^5)</td>
<td>Count of dist.(10^5)</td>
<td>Count of dist.(10^5)</td>
<td>Count of dist.(10^5)</td>
</tr>
<tr>
<td>1</td>
<td>0.940</td>
<td>0.931</td>
<td>0.922</td>
<td>0.920</td>
</tr>
<tr>
<td></td>
<td>115659</td>
<td>91792</td>
<td>6587</td>
<td>5217</td>
</tr>
<tr>
<td>2</td>
<td>0.942</td>
<td>0.943</td>
<td>0.920</td>
<td>0.918</td>
</tr>
<tr>
<td></td>
<td>92528</td>
<td>63512</td>
<td>6554</td>
<td>5479</td>
</tr>
<tr>
<td>3</td>
<td>0.956</td>
<td>0.935</td>
<td>0.930</td>
<td>0.920</td>
</tr>
<tr>
<td></td>
<td>122462</td>
<td>76873</td>
<td>6496</td>
<td>5312</td>
</tr>
<tr>
<td>4</td>
<td>0.951</td>
<td>0.928</td>
<td>0.928</td>
<td>0.918</td>
</tr>
<tr>
<td></td>
<td>103413</td>
<td>70582</td>
<td>6447</td>
<td>5373</td>
</tr>
<tr>
<td>5</td>
<td>0.946</td>
<td>0.949</td>
<td>0.921</td>
<td>0.918</td>
</tr>
<tr>
<td></td>
<td>111577</td>
<td>90014</td>
<td>6483</td>
<td>5374</td>
</tr>
<tr>
<td>6</td>
<td>0.960</td>
<td>0.936</td>
<td>0.922</td>
<td>0.920</td>
</tr>
<tr>
<td></td>
<td>108856</td>
<td>72069</td>
<td>6632</td>
<td>5407</td>
</tr>
<tr>
<td>7</td>
<td>0.972</td>
<td>0.936</td>
<td>0.920</td>
<td>0.920</td>
</tr>
<tr>
<td></td>
<td>77562</td>
<td>84914</td>
<td>6426</td>
<td>5509</td>
</tr>
<tr>
<td>8</td>
<td>0.944</td>
<td>0.934</td>
<td>0.920</td>
<td>0.918</td>
</tr>
<tr>
<td></td>
<td>93889</td>
<td>93092</td>
<td>6549</td>
<td>5413</td>
</tr>
<tr>
<td>9</td>
<td>0.954</td>
<td>0.930</td>
<td>0.923</td>
<td>0.918</td>
</tr>
<tr>
<td></td>
<td>84365</td>
<td>59242</td>
<td>6573</td>
<td>5426</td>
</tr>
<tr>
<td>10</td>
<td>0.942</td>
<td>0.931</td>
<td>0.923</td>
<td>0.920</td>
</tr>
<tr>
<td></td>
<td>84365</td>
<td>75053</td>
<td>6541</td>
<td>5171</td>
</tr>
<tr>
<td>Ave.</td>
<td>0.951</td>
<td>0.935</td>
<td>0.923</td>
<td>0.919</td>
</tr>
<tr>
<td></td>
<td>99467</td>
<td>78615</td>
<td>6529</td>
<td>5368</td>
</tr>
</tbody>
</table>

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

final temperature \(T_f\), the initial temperature \(T_0\), \(\eta\), the times of distance drops \(\text{dis}_\text{drop}, \text{per}\) and the total number of perturbations \(\text{total}_\text{per}\) 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
CHAPTER 3. SAMPLING SCHEMES FOR K-MEDOIDS ALGORITHM

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

<table>
<thead>
<tr>
<th>seed</th>
<th>CLARA</th>
<th>CLARANS</th>
<th>MCMRS</th>
<th>MCMRS-CLASA</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2.436</td>
<td>253</td>
<td>2.432</td>
<td>584</td>
</tr>
<tr>
<td>2</td>
<td>2.425</td>
<td>274</td>
<td>2.457</td>
<td>600</td>
</tr>
<tr>
<td>3</td>
<td>2.430</td>
<td>264</td>
<td>2.429</td>
<td>604</td>
</tr>
<tr>
<td>4</td>
<td>2.440</td>
<td>295</td>
<td>2.442</td>
<td>504</td>
</tr>
<tr>
<td>5</td>
<td>2.445</td>
<td>232</td>
<td>2.431</td>
<td>583</td>
</tr>
<tr>
<td>6</td>
<td>2.411</td>
<td>253</td>
<td>2.419</td>
<td>606</td>
</tr>
<tr>
<td>7</td>
<td>2.435</td>
<td>243</td>
<td>2.457</td>
<td>530</td>
</tr>
<tr>
<td>8</td>
<td>2.422</td>
<td>285</td>
<td>2.470</td>
<td>519</td>
</tr>
<tr>
<td>9</td>
<td>2.444</td>
<td>253</td>
<td>2.417</td>
<td>620</td>
</tr>
<tr>
<td>10</td>
<td>2.440</td>
<td>274</td>
<td>2.424</td>
<td>581</td>
</tr>
<tr>
<td>Ave.</td>
<td>2.429</td>
<td>263</td>
<td>2.438</td>
<td>573</td>
</tr>
</tbody>
</table>

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 $T_0$, $\eta$, 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>
<thead>
<tr>
<th>seed</th>
<th>CLARA</th>
<th>CLARANS</th>
<th>MCMRS</th>
<th>MCMRS-CLASA</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Average dist.</td>
<td>Count of dist.($10^5$)</td>
<td>Average dist.</td>
<td>Count of dist.($10^5$)</td>
</tr>
<tr>
<td>1</td>
<td>2829.56</td>
<td>30057</td>
<td>2816.43</td>
<td>79704</td>
</tr>
<tr>
<td>2</td>
<td>2833.36</td>
<td>31379</td>
<td>2831.77</td>
<td>56796</td>
</tr>
<tr>
<td>3</td>
<td>2839.22</td>
<td>32700</td>
<td>2806.48</td>
<td>76182</td>
</tr>
<tr>
<td>4</td>
<td>2821.52</td>
<td>36003</td>
<td>2814.16</td>
<td>59684</td>
</tr>
<tr>
<td>5</td>
<td>2827.43</td>
<td>28736</td>
<td>2821.41</td>
<td>44216</td>
</tr>
<tr>
<td>6</td>
<td>2842.22</td>
<td>28736</td>
<td>2822.10</td>
<td>54274</td>
</tr>
<tr>
<td>7</td>
<td>2838.23</td>
<td>30718</td>
<td>2817.13</td>
<td>55947</td>
</tr>
<tr>
<td>8</td>
<td>2843.91</td>
<td>28736</td>
<td>2819.38</td>
<td>56845</td>
</tr>
<tr>
<td>9</td>
<td>2820.76</td>
<td>24772</td>
<td>2823.58</td>
<td>64621</td>
</tr>
<tr>
<td>10</td>
<td>2819.52</td>
<td>28736</td>
<td>2826.95</td>
<td>59039</td>
</tr>
<tr>
<td>Ave.</td>
<td>2831.57</td>
<td>30057</td>
<td>2819.94</td>
<td>60731</td>
</tr>
</tbody>
</table>

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,
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.
CHAPTER 3. SAMPLING SCHEMES FOR K-MEDOIDS ALGORITHM

Get NumRun groups representative centroids

Calculate the nearest object to the centroid for each cluster

$n = 1$

Choose $NumRun = \frac{NumRun}{2}$ better groups

Sort $NumCandidate + NumSelect \times n$ nearest objects from the centroid in each cluster

$①$

Count = 1

Generate medoids from the nearest objects in each cluster

Count = Count + 1

Count = $NumSample + StepSample \times n$

Yes

Stop

No

$n = n + 1$

②

No

$NumRun = 1$

No

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 $512 \times 512$ 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

<table>
<thead>
<tr>
<th>seed</th>
<th>CLARA</th>
<th>CLARANS</th>
<th>MCMRS</th>
<th>IMCMRS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.228</td>
<td>2004</td>
<td>0.228</td>
<td>1609</td>
</tr>
<tr>
<td>2</td>
<td>0.221</td>
<td>2302</td>
<td>0.232</td>
<td>1076</td>
</tr>
<tr>
<td>3</td>
<td>0.236</td>
<td>2388</td>
<td>0.236</td>
<td>1351</td>
</tr>
<tr>
<td>4</td>
<td>0.230</td>
<td>2686</td>
<td>0.233</td>
<td>1154</td>
</tr>
<tr>
<td>5</td>
<td>0.227</td>
<td>2430</td>
<td>0.232</td>
<td>1572</td>
</tr>
<tr>
<td>6</td>
<td>0.237</td>
<td>2260</td>
<td>0.227</td>
<td>1223</td>
</tr>
<tr>
<td>7</td>
<td>0.232</td>
<td>2646</td>
<td>0.234</td>
<td>895</td>
</tr>
<tr>
<td>8</td>
<td>0.236</td>
<td>2516</td>
<td>0.231</td>
<td>1429</td>
</tr>
<tr>
<td>9</td>
<td>0.233</td>
<td>2345</td>
<td>0.231</td>
<td>1148</td>
</tr>
<tr>
<td>10</td>
<td>0.238</td>
<td>2728</td>
<td>0.231</td>
<td>1167</td>
</tr>
<tr>
<td>Ave.</td>
<td>0.232</td>
<td>2431</td>
<td>0.232</td>
<td>1140</td>
</tr>
</tbody>
</table>

**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.
CHAPTER 3. SAMPLING SCHEMES FOR K-MEDOIDS ALGORITHM

Table 3.6. Results of Experiment for Twelve Elliptic Clusters

<table>
<thead>
<tr>
<th>seed</th>
<th>CLARA</th>
<th>CLARANS</th>
<th>MCMRS</th>
<th>IMCMRS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.940</td>
<td>115659</td>
<td>0.931</td>
<td>91792</td>
</tr>
<tr>
<td>2</td>
<td>0.942</td>
<td>92528</td>
<td>0.943</td>
<td>63512</td>
</tr>
<tr>
<td>3</td>
<td>0.956</td>
<td>122462</td>
<td>0.935</td>
<td>76873</td>
</tr>
<tr>
<td>4</td>
<td>0.951</td>
<td>103413</td>
<td>0.928</td>
<td>70582</td>
</tr>
<tr>
<td>5</td>
<td>0.946</td>
<td>111577</td>
<td>0.949</td>
<td>90014</td>
</tr>
<tr>
<td>6</td>
<td>0.960</td>
<td>108856</td>
<td>0.936</td>
<td>72069</td>
</tr>
<tr>
<td>7</td>
<td>0.972</td>
<td>77362</td>
<td>0.936</td>
<td>84914</td>
</tr>
<tr>
<td>8</td>
<td>0.944</td>
<td>93889</td>
<td>0.934</td>
<td>93092</td>
</tr>
<tr>
<td>9</td>
<td>0.954</td>
<td>84365</td>
<td>0.930</td>
<td>59242</td>
</tr>
<tr>
<td>10</td>
<td>0.942</td>
<td>84365</td>
<td>0.931</td>
<td>75053</td>
</tr>
<tr>
<td>Ave.</td>
<td>0.951</td>
<td>99467</td>
<td>0.935</td>
<td>78615</td>
</tr>
</tbody>
</table>

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
CHAPTER 3. SAMPLING SCHEMES FOR K-MEDOIDS ALGORITHM

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

<table>
<thead>
<tr>
<th>seed</th>
<th>CLARA</th>
<th>CLARANS</th>
<th>MCMRS</th>
<th>IMCMRS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2.436</td>
<td>253</td>
<td>2.432</td>
<td>584</td>
</tr>
<tr>
<td>2</td>
<td>2.425</td>
<td>274</td>
<td>2.457</td>
<td>600</td>
</tr>
<tr>
<td>3</td>
<td>2.430</td>
<td>264</td>
<td>2.429</td>
<td>604</td>
</tr>
<tr>
<td>4</td>
<td>2.440</td>
<td>295</td>
<td>2.442</td>
<td>504</td>
</tr>
<tr>
<td>5</td>
<td>2.405</td>
<td>232</td>
<td>2.431</td>
<td>583</td>
</tr>
<tr>
<td>6</td>
<td>2.411</td>
<td>253</td>
<td>2.419</td>
<td>606</td>
</tr>
<tr>
<td>7</td>
<td>2.435</td>
<td>243</td>
<td>2.457</td>
<td>530</td>
</tr>
<tr>
<td>8</td>
<td>2.422</td>
<td>285</td>
<td>2.470</td>
<td>519</td>
</tr>
<tr>
<td>9</td>
<td>2.444</td>
<td>253</td>
<td>2.417</td>
<td>620</td>
</tr>
<tr>
<td>10</td>
<td>2.440</td>
<td>274</td>
<td>2.424</td>
<td>581</td>
</tr>
<tr>
<td>Ave.</td>
<td>2.429</td>
<td>263</td>
<td>2.438</td>
<td>573</td>
</tr>
</tbody>
</table>
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>
<thead>
<tr>
<th>seed</th>
<th>CLARA</th>
<th>CLARANS</th>
<th>MCMRS</th>
<th>IMCMRS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Average dist.</td>
<td>Count of dist.($10^5$)</td>
<td>Average dist.</td>
<td>Count of dist.($10^5$)</td>
</tr>
<tr>
<td>1</td>
<td>4.559</td>
<td>154809</td>
<td>4.432</td>
<td>37604</td>
</tr>
<tr>
<td>2</td>
<td>4.592</td>
<td>163692</td>
<td>4.359</td>
<td>53234</td>
</tr>
<tr>
<td>3</td>
<td>4.551</td>
<td>178918</td>
<td>4.381</td>
<td>37512</td>
</tr>
<tr>
<td>4</td>
<td>4.578</td>
<td>172574</td>
<td>4.398</td>
<td>40559</td>
</tr>
<tr>
<td>5</td>
<td>4.559</td>
<td>182725</td>
<td>4.367</td>
<td>39694</td>
</tr>
<tr>
<td>6</td>
<td>4.526</td>
<td>167498</td>
<td>4.384</td>
<td>41370</td>
</tr>
<tr>
<td>7</td>
<td>4.527</td>
<td>185263</td>
<td>4.380</td>
<td>32312</td>
</tr>
<tr>
<td>8</td>
<td>4.483</td>
<td>162423</td>
<td>4.394</td>
<td>39600</td>
</tr>
<tr>
<td>9</td>
<td>4.545</td>
<td>190338</td>
<td>4.377</td>
<td>36707</td>
</tr>
<tr>
<td>10</td>
<td>4.514</td>
<td>180187</td>
<td>4.406</td>
<td>35835</td>
</tr>
<tr>
<td>Ave.</td>
<td>4.543</td>
<td>173843</td>
<td>4.388</td>
<td>39443</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>seed</th>
<th>CLARA</th>
<th>CLARANS</th>
<th>MCMRS</th>
<th>IMCMRS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Average</td>
<td>Count of</td>
<td>Average</td>
<td>Count of</td>
</tr>
<tr>
<td></td>
<td>dist.</td>
<td>dist. $(10^5)$</td>
<td>dist.</td>
<td>dist. $(10^5)$</td>
</tr>
<tr>
<td>1</td>
<td>2829.56</td>
<td>30057</td>
<td>2816.43</td>
<td>79704</td>
</tr>
<tr>
<td>2</td>
<td>2833.36</td>
<td>31379</td>
<td>2831.77</td>
<td>56796</td>
</tr>
<tr>
<td>3</td>
<td>2839.22</td>
<td>32700</td>
<td>2806.48</td>
<td>76182</td>
</tr>
<tr>
<td>4</td>
<td>2821.52</td>
<td>36003</td>
<td>2814.16</td>
<td>50684</td>
</tr>
<tr>
<td>5</td>
<td>2827.43</td>
<td>30718</td>
<td>2804.41</td>
<td>42164</td>
</tr>
<tr>
<td>6</td>
<td>2842.22</td>
<td>32700</td>
<td>2822.10</td>
<td>54274</td>
</tr>
<tr>
<td>7</td>
<td>2838.23</td>
<td>30718</td>
<td>2817.13</td>
<td>55947</td>
</tr>
<tr>
<td>8</td>
<td>2843.91</td>
<td>30718</td>
<td>2819.38</td>
<td>56845</td>
</tr>
<tr>
<td>9</td>
<td>2820.76</td>
<td>24772</td>
<td>2823.58</td>
<td>64621</td>
</tr>
<tr>
<td>10</td>
<td>2819.52</td>
<td>28736</td>
<td>2826.95</td>
<td>59039</td>
</tr>
<tr>
<td>Ave.</td>
<td>2831.57</td>
<td>30057</td>
<td>2819.94</td>
<td>60731</td>
</tr>
</tbody>
</table>

**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 is 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

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

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. $\epsilon$ 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 $\Delta D$ is defined as the distortion of current clusters minus the distortion of previous clusters. The perturbation is accepted if $e^{-\frac{\Delta D}{T_n}} > \gamma$, where $\gamma$ 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 $i$th 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
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 perturbation 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 \( C^{(1)}_i, i = 1, 2, \ldots N \), \( N \) is the number of clusters. Set iterative number \( n = 1 \) and \( D_0 = \infty \).

Step 2. Assign the training data \( X_p \) to the \( i \)th 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 \( i \)th cluster.

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

\( \delta_i(T_n) \) is a perturbation 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 \( \sigma^2 \alpha^n, \alpha = 0.95, \sigma^2 \) 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 $C_{\text{init}}$ using GLA algorithm. Set $C_{\text{curr}}=C_{\text{best}}=C_{\text{init}}$.

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 $C_{\text{curr}}$ with probability $P_i$. For each test solution, if the $q$th element of the test solution is changed and the value of $Count_q$ is smaller than $\nu$, then increment $Count_q$. If all the values of the counters
are equal to \( \nu \), 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 \( D(C_{\text{best}}) > D(C_i) \), set \( C_{\text{curr}} = C_i \) and go to step 6; otherwise go to next step.

**Step 5.** Calculate \( \Delta D = D(C_{\text{curr}}) - D(C_i) \) and set \( \hat{T}_n = \hat{T}_0 \alpha^n \). Generate a random number \( \gamma \) (0 \( \leq \) \( \gamma \) \( \leq \) 1), if \( \gamma < e^{\frac{\Delta D}{\hat{T}_n}} \), then set \( C_{\text{curr}} = C_i \) 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 \( D(C_{\text{best}}) \) remains unchanged for \( M \) iterations, then set \( C_{\text{curr}} = C_{\text{best}} \), clear the counter \( Count_j, j = 1, 2, \ldots T \). If \( D(C_{\text{best}}) > D(C_{\text{curr}}) \) then set \( C_{\text{best}} = C_{\text{curr}} \). 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 \( \text{TABU-GLA} \)), and our proposed tabu search approach with simulated annealing (referred to as \( \text{TABU-SA} \)) (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
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 \( i \)th cluster. \( T \) is the total number of training patterns and \( k \) is the number of dimension for each pattern. \( X_j^{(i)} \) is the \( j \)th pattern belonging to the \( i \)th cluster.

Three 512 \( \times \) 512 images, the "Lena", "Baboo", and "Pepper" images were
used as the test material. $4 \times 4$ 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 $\nu$, the limit number of the same current best iterations, the initial temperature $T_0$ and the value of $\alpha$ were $20$, $16384$, $100$, $3$, $30$, $500$ 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

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

Preliminary experimental results demonstrate the proposed tabu search ap-
Table 4.2: Performance Comparison Using *Baboo* Image as The Training Patterns

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

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

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>TS-SA with GLA initialization</th>
<th>TS-SA with random initialization</th>
<th>GLA</th>
<th>TS-GLA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initialization</td>
<td>127.286</td>
<td>140.189</td>
<td>82.332</td>
<td>75.351</td>
</tr>
<tr>
<td>1 hour</td>
<td>60.328</td>
<td>61.788</td>
<td>66.788</td>
<td>67.041</td>
</tr>
<tr>
<td>2 hour</td>
<td>60.117</td>
<td>60.891</td>
<td>66.788</td>
<td>66.005</td>
</tr>
<tr>
<td>3 hour</td>
<td>60.117</td>
<td>60.891</td>
<td>66.788</td>
<td>64.969</td>
</tr>
<tr>
<td>4 hour</td>
<td>60.117</td>
<td>60.891</td>
<td>66.788</td>
<td>64.409</td>
</tr>
<tr>
<td>5 hour</td>
<td>60.117</td>
<td>60.891</td>
<td>66.788</td>
<td>64.185</td>
</tr>
<tr>
<td>6 hour</td>
<td>60.117</td>
<td>60.891</td>
<td>66.788</td>
<td>63.804</td>
</tr>
<tr>
<td>7 hour</td>
<td>60.117</td>
<td>60.891</td>
<td>66.788</td>
<td>63.496</td>
</tr>
<tr>
<td>8 hour</td>
<td>60.117</td>
<td>60.891</td>
<td>66.788</td>
<td>63.381</td>
</tr>
</tbody>
</table>

The approach with simulated annealing algorithm for cluster generation is superior to the tabu search approach with Generalised Lloyd algorithm (GLA).
**CHAPTER 4. IMPROVED CENTROID-BASED CLUSTERING ALGORITHMS**

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

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

with a much smaller set of codevectors

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

, 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

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

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

$$i = \text{argmin}_p D(X, C_p). \quad (4.2)$$
CHAPTER 4. IMPROVED CENTROID-BASED CLUSTERING ALGORITHMS

112

Figure 4.10. Block Diagram of VQ Compression

Assuming $\hat{X}_j$ 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:

\[ MSE = \frac{\sum_{j=1}^{T} D(X_j, \hat{X}_j)}{T} \]  
\[ SNR = \frac{\sum_{j=1}^{T} X_j^2}{\sum_{j=1}^{T} D(X_j, \hat{X}_j)} \]  
\[ PSNR = 10 \log \left( \frac{(peak \ signal)^2}{MSE} \right) \]

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 $X - m \cdot \bar{1} = \{x_1 - m, x_2 - m, \ldots, x_k - m\}$
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 $\hat{X} = \{M_i + r_{j_1}, M_i + r_{j_2}, \ldots, M_i + r_{j_k}\}$ 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. 1996b), 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
CHAPTER 4. IMPROVED CENTROID-BASED CLUSTERING ALGORITHMS

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

Figure 4.11: Schematic for Mean-Residual Vector Quantization ($M/R VQ$)
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 \( X_i = \{x_{i1}, x_{i2}, \ldots, x_{ik}\} \), \( i = 1, 2, \ldots, T, j = 1, 2, \ldots, k \), \( T \) 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 MaxIt 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

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

\( T \), \( X_i \) 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. $P_m \times (\text{population size}) \times (\text{codebook size})$ individuals are mutated by adding the random value to the mean code scalar and residual code vector, where the random value can be set between $\pm (\alpha^{Gen} \times \frac{\text{deviation of the gene}}{2})$, 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 $(M/R VQ)$
4.3.2 Experiments

The test materials for these experiments consisted of two gray-level images, i.e., Lena and Pepper images with resolution $512 \times 512$ 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 GLAMaxIte. 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 GLAMaxIte, 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 GLAMaxIte, 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
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

Parameters 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 $\epsilon$ 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 $GLA$

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

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

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

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

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

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

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

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

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

<table>
<thead>
<tr>
<th>Seed</th>
<th>GA (dB)</th>
<th>GLA (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>34.3471</td>
<td>33.7332</td>
</tr>
<tr>
<td>2</td>
<td>34.3842</td>
<td>33.7675</td>
</tr>
<tr>
<td>3</td>
<td>34.4132</td>
<td>33.7442</td>
</tr>
<tr>
<td>4</td>
<td>34.3103</td>
<td>33.8159</td>
</tr>
<tr>
<td>5</td>
<td>34.3240</td>
<td>33.6789</td>
</tr>
<tr>
<td>6</td>
<td>34.3824</td>
<td>33.7931</td>
</tr>
<tr>
<td>7</td>
<td>34.3406</td>
<td>33.7165</td>
</tr>
<tr>
<td>8</td>
<td>34.4045</td>
<td>33.7849</td>
</tr>
<tr>
<td>9</td>
<td>34.4379</td>
<td>33.8309</td>
</tr>
<tr>
<td>10</td>
<td>34.4055</td>
<td>33.7759</td>
</tr>
<tr>
<td>Ave.</td>
<td>34.3750</td>
<td>33.7641</td>
</tr>
</tbody>
</table>
Table 4.9: Performance Comparison for M/R VQ with Mean Codebook Size 256 and Residual Codebook Size 256 Using Lena Image

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

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 $\epsilon$ 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

<table>
<thead>
<tr>
<th>Test Image</th>
<th>GA (dB)</th>
<th>GLA (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>F16</td>
<td>29.7143</td>
<td>29.5162</td>
</tr>
<tr>
<td>Pepper</td>
<td>32.3492</td>
<td>32.1740</td>
</tr>
<tr>
<td>Lena</td>
<td>32.5417</td>
<td>32.4112</td>
</tr>
</tbody>
</table>

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:

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

where $X^p_i$ is the $p$th 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.

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

(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
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 $\epsilon$

Figure 4.19. Pseudo Code for $K$-Means Method

Random Initialization (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 split into two smaller ones $C_1$ and $C_2$, where $C_1 = C \times (1 + \delta)$, $C_2 = C \times (1 - \delta)$, $\delta$ 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
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 splitting 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.
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)

<table>
<thead>
<tr>
<th>Number of Sample Data</th>
<th>Random Initialization</th>
<th>Binary Splitting</th>
<th>Incremental Splitting</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>0.153637</td>
<td>0.137252</td>
<td>0.131204</td>
</tr>
<tr>
<td>2000</td>
<td>0.176120</td>
<td>0.160780</td>
<td>0.155279</td>
</tr>
<tr>
<td>4000</td>
<td>0.184560</td>
<td>0.175639</td>
<td>0.169673</td>
</tr>
<tr>
<td>8000</td>
<td>0.190036</td>
<td>0.186481</td>
<td>0.179899</td>
</tr>
<tr>
<td>16000</td>
<td>0.194979</td>
<td>0.193206</td>
<td>0.186768</td>
</tr>
<tr>
<td>32000</td>
<td>0.196600</td>
<td>0.197891</td>
<td>0.192024</td>
</tr>
</tbody>
</table>
Table 4.12: Error for Different Number of Clusters (Sample Size = 10000)

<table>
<thead>
<tr>
<th>Number of Clusters</th>
<th>Random Initialization</th>
<th>Binary Splitting</th>
<th>Incremental Splitting</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>0.536971</td>
<td>0.548424</td>
<td>0.535818</td>
</tr>
<tr>
<td>64</td>
<td>0.386860</td>
<td>0.391035</td>
<td>0.379582</td>
</tr>
<tr>
<td>128</td>
<td>0.272711</td>
<td>0.275201</td>
<td>0.265303</td>
</tr>
<tr>
<td>256</td>
<td>0.191920</td>
<td>0.188622</td>
<td>0.182477</td>
</tr>
<tr>
<td>512</td>
<td>0.134390</td>
<td>0.129469</td>
<td>0.1123353</td>
</tr>
<tr>
<td>1024</td>
<td>0.092086</td>
<td>0.084134</td>
<td>0.080145</td>
</tr>
</tbody>
</table>

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 2004a, 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. 2000c, 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. 2000c, 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 $\mathbb{R}^k$ 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.
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
CHAPTER 4. IMPROVED CENTROID-BASED CLUSTERING ALGORITHMS

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 $512 \times 512$ and a binary watermark $W$ of size $128 \times 128$ as shown in Figure 4.25. The Lena image is divided into 16384 blocks of size $4 \times 4$, 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

$$NHS = 1 - \frac{HD(W, W')}{\text{number of watermark bits}}$$  \hspace{1cm} (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.

<table>
<thead>
<tr>
<th>Number of Clusters</th>
<th>32</th>
<th>64</th>
<th>128</th>
<th>256</th>
<th>512</th>
<th>1024</th>
<th>2048</th>
<th>4096</th>
<th>8192</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSNR of the watermarked image (dB)</td>
<td>24.37</td>
<td>25.93</td>
<td>27.63</td>
<td>28.76</td>
<td>29.80</td>
<td>30.63</td>
<td>31.11</td>
<td>31.99</td>
<td>32.47</td>
</tr>
</tbody>
</table>

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 ~ 4.30 show the watermarked images and the corresponding
Table 4.14. *NHS* Value for Various Attacks

<table>
<thead>
<tr>
<th>Attack</th>
<th>NHS</th>
</tr>
</thead>
<tbody>
<tr>
<td>No Attack</td>
<td>1.0</td>
</tr>
<tr>
<td>JPEG (QF=100%)</td>
<td>0.999</td>
</tr>
<tr>
<td>JPEG (QF=80%)</td>
<td>0.962</td>
</tr>
<tr>
<td>VQ (Codebook 1)</td>
<td>1.0</td>
</tr>
<tr>
<td>VQ (Codebook 2)</td>
<td>0.778</td>
</tr>
<tr>
<td>Median Filter (radius=1)</td>
<td>0.835</td>
</tr>
<tr>
<td>Blurring (radius=1, threshold=10)</td>
<td>0.872</td>
</tr>
<tr>
<td>Sharpen</td>
<td>0.969</td>
</tr>
<tr>
<td>Contrast Enhancement 4%</td>
<td>0.904</td>
</tr>
<tr>
<td>Image Cropping in The Upper-Left Corner</td>
<td>0.849</td>
</tr>
</tbody>
</table>

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, \( C = y_1, y_2, \ldots, y_N \), 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 \( y_j = (y_{j1}, y_{j2}, \ldots, y_{jk}) \) in the codebook \( C \) is found for each input vector \( x = (x_1, x_2, \ldots, x_k) \) 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 IEEENNS 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. 1996c, Pan et al. 1996b). 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 IEEENNS, four extra inequalities are used to reject those codewords which have not been rejected by IEEENNS. 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 2002a) 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. 2000a) 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:

$$S_x = \sum_{l=1}^{k} x_l$$  \hspace{1cm} (4.8)

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

$$m_x = \frac{\sum_{l=1}^{k} x_l}{k}$$  \hspace{1cm} (4.9)

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

$$S_x = km_x$$  \hspace{1cm} (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.
CHAPTER 4. IMPROVED CENTROID-BASED CLUSTERING ALGORITHMS

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

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

$$\|x\| = \sqrt{\sum_{l=1}^{k} x_l^2}$$  \hspace{1cm} (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:

$$v_x^2 = \|x\|^2 - k \cdot m_x^2$$  \hspace{1cm} (4.13)

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

$$d_{\min} = \min\{d(x, y_i) | y_i \text{ has been inspected}\}$$  \hspace{1cm} (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:

$$\sum_{l=1}^{q} (x_l - y_{il})^2 \geq d_{\min}$$  \hspace{1cm} (4.15)
which guarantees that $d(x, y_i) \geq d_{\text{min}}$, 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 $d_{\text{min}}$, if the mean of the uninspected codeword $y_i$ satisfies:
\[
m_i \geq m_x + \sqrt{\frac{d_{\text{min}}}{k}} \quad \text{or} \quad m_i \leq m_x - \sqrt{\frac{d_{\text{min}}}{k}} \tag{4.17}
\]

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 $m_{\text{max}} = m_x + \sqrt{\frac{d_{\text{min}}}{k}}$ and $m_{\text{min}} = m_x - \sqrt{\frac{d_{\text{min}}}{k}}$. In other words, the search area is bounded by two lines $l_1$ and $l_2$ perpendicular to the central line $l$ at $L_{\text{max}} = (m_{\text{max}}, m_{\text{max}}, \cdots, m_{\text{max}})$ and $L_{\text{min}} = (m_{\text{min}}, m_{\text{min}}, \cdots, m_{\text{min}})$, 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 $d_{\text{min}}$ between the input vector and this tentative matching codeword is calculated. Then the codewords $y_i$ for which $m_i \geq m_x + \sqrt{\frac{d_{\text{min}}}{k}}$ or $m_i \leq m_x - \sqrt{\frac{d_{\text{min}}}{k}}$ are eliminated. Otherwise, the PDS is applied to calculate the distortion and update $d_{\text{min}}$. 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 Error Inequality (IAEI) criterion (Pan et al. 1996a, Pan et al. 1996b). 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

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

or

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

then $d(x, y_i) \geq d_{\text{min}}$, 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,
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

$$ (v_x - v_i)^2 \leq d(x, y_i) \quad (4.20) $$

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 $d_{\text{min}}$, if the variance of the uninspected codeword $y_i$ satisfies

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

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 $m_i \geq m_x + \frac{d_{\text{min}}}{k} \text{ or } m_i \leq m_x - \frac{d_{\text{min}}}{k}$, then the codeword $y_i$ can be rejected. Otherwise, in the second step, if $v_i \geq v_x + \sqrt{d_{\text{min}}} \text{ or } v_i \leq v_x - \sqrt{d_{\text{min}}}$, 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:**

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

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 $d_{\text{min}}$, if the mean and variance of the uninspected codeword $y_i$ satisfy:

$$k \cdot (m_x - m_i)^2 + (v_x - v_i)^2 \geq d_{\text{min}}$$  \hspace{1cm} (4.23)

then $d(x, y_i) \geq d_{\text{min}}$, 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 $x_f$, $y_i f$, $x_s$, and $y_i s$ respectively, and $v_{xf}$, $v_{if}$, $v_{xs}$, and $v_{is}$ are the variances of $x_f$, $y_i f$, $x_s$, and $y_i s$ respectively, if

$$(S_{xf} - S_{if})^2 + \frac{k \cdot (v_{xf} - v_{if})^2}{2} \geq \frac{k \cdot d_{\text{min}}}{2}$$  \hspace{1cm} (4.24)

or

$$(S_{xs} - S_{is})^2 + \frac{k \cdot (v_{xs} - v_{is})^2}{2} \geq \frac{k \cdot d_{\text{min}}}{2}$$  \hspace{1cm} (4.25)

then $D(x, y_i) \geq d_{\text{min}}$, 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

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

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

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 \(d_{1\text{min}}\), if the uninspected codeword \(y_i\) satisfies

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

then \(d_1(x, y_i) \geq d_{1\text{min}}\) 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

\[
(\|x\| - \|y_i\|)^2 \leq d(x, y_i)
\]

(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 \( d_{\text{min}} \), if the norm \( \|y_i\| \) of the uninspected codeword \( y_i \) satisfies:

\[
\|y_i\| \geq \|x\| + \sqrt{d_{\text{min}}} \quad \text{or} \quad \|y_i\| \leq \|x\| - \sqrt{d_{\text{min}}}
\]

(4.32)

Then \( y_i \) will not be the nearest neighbor to \( x \). The elimination process of the \textit{EEENNS} algorithm consists of three steps. In the first step, if \( m_i \geq m_x + \sqrt{d_{\text{min}}} \) or \( m_i \leq m_x - \sqrt{d_{\text{min}}} \), then the codeword \( y_i \) can be rejected. Or else, in the second step, \( v_i \geq v_x + \sqrt{d_{\text{min}}} \) or \( v_i \leq v_x - \sqrt{d_{\text{min}}} \), then the codeword \( y_i \) can be rejected. Otherwise, in the third step, if \( \|y_i\| \geq \|x\| + \sqrt{d_{\text{min}}} \) or \( \|y_i\| \leq \|x\| - \sqrt{d_{\text{min}}} \), 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 \textit{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 \textit{HTPDS} (Lu et al. 2000a), \textit{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:  
\[ H_1 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \text{ and } H_{n+1} = \begin{bmatrix} H_n & H_n \\ H_n & -H_n \end{bmatrix}. \]

Definition 8: The Hadamard-transformed vector \( X \) of the vector \( x \) is defined as:
\[ X = H_n x \quad (4.33) \]

Definition 9: The Hadamard-transformed variance of vector \( X \) can be defined as:
\[ V_X = \sqrt{\sum_{l=2}^{k} X_l^2} \quad (4.34) \]

Definition 10: The Hadamard-transformed norm of vector \( X \) can be defined as:
\[ \| X \| = \sqrt{\sum_{l=1}^{k} X_l^2} \quad (4.35) \]

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:
\[ d(X, Y_i) = kd(x, y_i) \quad (4.36) \]

Proof:
\[ d(X, Y_i) = \sum_{l=1}^{k} (X_l - Y_i)^2 = (X - Y_i)^T (X - Y_i) = [H_n(x - y_i)]^T [H_n(x - y_i)] = (x - y_i)^T H_n^T H_n (x - y_i) = (x - y_i)^T k I_k (x - y_i) = k (x - y_i)^T (x - y_i) = k \sum_{l=1}^{k} (x_l - y_i)^2 = kd(x, y_i) \]

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.,
\[ X_1 = S_x \quad (4.37) \]
CHAPTER 4. IMPROVED CENTROID-BASED CLUSTERING ALGORITHMS

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:

$$V_X = \sqrt{\|X\|^2 - X_1^2}$$  \hspace{1cm} (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:

$$V_X = \sqrt{k \cdot v_x}$$  \hspace{1cm} (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

$$(X_1 - Y_{il})^2 + (V_X - V_i)^2 \leq \sqrt{d(X,Y_i)}$$  \hspace{1cm} (4.40)

Proof: This inequality is equivalent to the following inequalities:

$$\Leftrightarrow \quad X_1^2 + Y_{il}^2 - 2X_1Y_{il} + V_1^2 - 2V_XV_i \leq \sum_{l=1}^k (X_l - Y_{il})^2$$
$$\Leftrightarrow \quad (X_1^2 + V_1^2) + (Y_{il}^2 + V_i^2) - 2X_1Y_{il} - 2V_XV_i \leq \sum_{l=1}^k (X_l - Y_{il})^2$$
$$\Leftrightarrow \quad \sum_{l=1}^k X_l^2 + \sum_{l=1}^k Y_{il}^2 - 2X_1Y_{il} - 2V_XV_i \leq \sum_{l=1}^k X_l^2 + \sum_{l=1}^k Y_{il}^2 - \sum_{l=1}^k 2X_lY_{il}$$
$$\Leftrightarrow \quad -2V_XV_i \leq -\sum_{l=2}^k 2X_lY_{il}$$
$$\Leftrightarrow \quad V_XV_i \geq \sum_{l=2}^k X_lY_{il}$$
$$\Leftrightarrow \quad \sqrt{\sum_{l=2}^k X_l^2 \cdot \sum_{l=2}^k Y_{il}^2} \geq \sum_{l=2}^k X_lY_{il}$$

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:

$$|X_1 - Y_{il}| \leq \sqrt{d(X,Y_i)}$$  \hspace{1cm} (4.41)
In fact, since \((X_1 - Y_{i1})^2\) is one of the summation items in \(\sum_{i=1}^{k}(X_i - Y_{i})^2\), above inequality is obviously tenable. **Corollary 2:**

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

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

\[ ||\|X\| - \|Y_i\|| \leq \sqrt{d(X, Y_i)} \]  (4.43)

**Proof:** This inequality is equivalent to the following inequalities:

\[
\Leftrightarrow \quad \|X\|^2 + \|Y_i\|^2 - 2\|X\| \cdot \|Y_i\| \leq \sum_{i=1}^{k}(X_i - Y_{i})^2 \\
\Leftrightarrow \quad \sum_{i=1}^{k} X_i^2 + \sum_{i=1}^{k} Y_{i}^2 - 2\|X\| \cdot \|Y_i\| \leq \sum_{i=1}^{k} X_i^2 + \sum_{i=1}^{k} Y_{i}^2 - \sum_{i=1}^{k} 2X_i Y_{i} \\
\Leftrightarrow \quad \frac{2\|X\| \cdot \|Y_i\|}{\|Y_i\|} \geq \sum_{i=1}^{k} 2X_i Y_{i} \\
\Leftrightarrow \quad \sqrt{\sum_{i=1}^{k} X_i^2} \cdot \sqrt{\sum_{i=1}^{k} Y_{i}^2} \geq \sum_{i=1}^{k} X_i Y_{i}
\]

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. 2000a) 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 $D_{\text{min}}$, if the first element $Y_{il}$ of the uninspected transformed codeword $Y_i$ is larger than $\text{MAXSUM} = X_1 + \sqrt{D_{\text{min}}}$ or less than $\text{MINSUM} = X_1 - \sqrt{D_{\text{min}}}$, 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 $\text{MINSUM}$ to $\text{MAXSUM}$. 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 $\|\|Y_i\| - \|X\|\| \geq \sqrt{D_{\text{min}}}$ 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 $\|\|Y_i\| - \|X\|\| \geq \sqrt{D_{\text{min}}}$ 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 $D_{\text{min}}$, four elimination criteria based on transformed vector $X$ and codeword $Y_i$ can be stated as follows:
Theorem 10: If

\[ Y_{i1} \geq X_1 + \sqrt{D_{\text{min}}} \quad \text{or} \quad Y_{i1} \leq X_1 - \sqrt{D_{\text{min}}} \]  \hspace{1cm} (4.44)

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

Theorem 11: If

\[ V_i \geq V_X + \sqrt{D_{\text{min}}} \quad \text{or} \quad V_i \leq V_X - \sqrt{D_{\text{min}}} \]  \hspace{1cm} (4.45)

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

Theorem 12: If

\[ (X_1 - Y_{i1})^2 + (V_X - V_i)^2 \geq \sqrt{D_{\text{min}}} \]  \hspace{1cm} (4.46)

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

Theorem 13: If

\[ \|Y_i\| \geq \|X\| + \sqrt{D_{\text{min}}} \quad \text{or} \quad \|Y_i\| \leq \|X\| - \sqrt{D_{\text{min}}} \]  \hspace{1cm} (4.47)

Then \( d(X, Y_i) \geq D_{\text{min}} \), 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 \textit{HTEEENNS} (Chu, Roddick, Lu & Pan 2004b). Because our algorithm is based on four elimination criterion, so we can denote our algorithm as the improved \textit{HTEEENNS}, i.e., \textit{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 $p = \arg\min_i |X_1 - Y_{i1}|$, 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 × 8 = 64 or 16 × 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 HTEENNS and IHTEENNS 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. 2000a) 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

<table>
<thead>
<tr>
<th>Codebook size</th>
<th>512</th>
<th>1024</th>
</tr>
</thead>
<tbody>
<tr>
<td>Performance</td>
<td>CPU Time(s)</td>
<td>Complexity</td>
</tr>
<tr>
<td>Dimension</td>
<td>8 × 8</td>
<td>16 × 16</td>
</tr>
<tr>
<td>FS</td>
<td>9.33</td>
<td>8.64</td>
</tr>
<tr>
<td>PDS (Bei &amp; Gray 1985)</td>
<td>1.69</td>
<td>1.87</td>
</tr>
<tr>
<td>ENNS (Guan &amp; Kamel 1992)</td>
<td>0.31</td>
<td>0.47</td>
</tr>
<tr>
<td>IENNS (Pan &amp; Huang 1998)</td>
<td>0.21</td>
<td>0.40</td>
</tr>
<tr>
<td>EENNS (Lee &amp; Chen 1994)</td>
<td>0.24</td>
<td>0.42</td>
</tr>
<tr>
<td>IEEENNS (Baek et al. 1997)</td>
<td>0.22</td>
<td>0.36</td>
</tr>
<tr>
<td>EENNS (Lu &amp; Sun 2003)</td>
<td>0.24</td>
<td>0.40</td>
</tr>
<tr>
<td>SVEENNS (Pan et al. 2003)</td>
<td>0.22</td>
<td>0.34</td>
</tr>
<tr>
<td>NOS (Wu &amp; Lin 2000)</td>
<td>0.76</td>
<td>0.84</td>
</tr>
<tr>
<td>TNOS (Jiang et al. 2003)</td>
<td>0.29</td>
<td>0.32</td>
</tr>
<tr>
<td>HTPDS (Lu et al. 2000a)</td>
<td>0.20</td>
<td>0.26</td>
</tr>
<tr>
<td>Proposed HTEEENNS</td>
<td>0.18</td>
<td>0.24</td>
</tr>
<tr>
<td>Proposed IHTEEENNS</td>
<td>0.18</td>
<td>0.24</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Codebook size</th>
<th>512</th>
<th>1024</th>
</tr>
</thead>
<tbody>
<tr>
<td>Performance</td>
<td>CPU Time(s)</td>
<td>Complexity</td>
</tr>
<tr>
<td>Dimension</td>
<td>8 × 8</td>
<td>16 × 16</td>
</tr>
<tr>
<td>FS</td>
<td>8.75</td>
<td>9.17</td>
</tr>
<tr>
<td>PDS (Bei &amp; Gray 1985)</td>
<td>3.61</td>
<td>4.08</td>
</tr>
<tr>
<td>ENNS (Guan &amp; Kamel 1992)</td>
<td>1.30</td>
<td>1.64</td>
</tr>
<tr>
<td>IENNS (Pan &amp; Huang 1998)</td>
<td>0.87</td>
<td>1.57</td>
</tr>
<tr>
<td>EENNS (Lee &amp; Chen 1994)</td>
<td>1.17</td>
<td>1.62</td>
</tr>
<tr>
<td>IEEENNS (Baek et al. 1997)</td>
<td>0.89</td>
<td>1.26</td>
</tr>
<tr>
<td>EENNS (Lu &amp; Sun 2003)</td>
<td>1.11</td>
<td>1.54</td>
</tr>
<tr>
<td>SVEENNS (Pan et al. 2003)</td>
<td>0.83</td>
<td>1.13</td>
</tr>
<tr>
<td>NOS (Wu &amp; Lin 2000)</td>
<td>2.01</td>
<td>2.31</td>
</tr>
<tr>
<td>TNOS (Jiang et al. 2003)</td>
<td>1.08</td>
<td>1.23</td>
</tr>
<tr>
<td>HTPDS (Lu et al. 2000a)</td>
<td>0.87</td>
<td>1.13</td>
</tr>
<tr>
<td>Proposed HTEEENNS</td>
<td>0.77</td>
<td>1.00</td>
</tr>
<tr>
<td>Proposed IHTEEENNS</td>
<td>0.65</td>
<td>0.85</td>
</tr>
</tbody>
</table>
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 $f(\cdot)$, the $i$th particle for the $t$th iteration can be represented as

$$X^t_i = (x^t_i(1), x^t_i(2), \ldots, x^t_i(D)). \quad (5.1)$$

Assume that the best previous position of the $i$th particle at the $t$th iteration is represented as

$$P^t_i = (p^t_i(1), p^t_i(2), \ldots, p^t_i(D)), \quad (5.2)$$

then

$$f(P^t_i) \leq f(P^{t-1}_i) \leq \cdots \leq f(P^1_i). \quad (5.3)$$

The velocity of the $i$th particle at the $t$th iteration can be represented as

$$V^t_i = (v^t_i(1), v^t_i(2), \ldots, v^t_i(D)). \quad (5.4)$$

$$G^t = (X^t(1), X^t(2), \ldots, X^t(D)) \quad (5.5)$$

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

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

$$V^{t+1}_i = V^t_i + C_1 r_1 (P^t_i - X^t_i) + C_2 r_2 (G^t - X^t_i) \quad (5.6)$$

$$X^{t+1}_i = X^t_i + V^{t+1}_i, \quad i = 0, \ldots, N - 1 \quad (5.7)$$

where $N$ is particle size, $-V_{max} \leq V^{t+1}_i \leq V_{max}$, ($V_{max}$ 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 \(t^{th}\) 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.

\[
F_{2}(x, y) = -xsin(\sqrt{|x|}) - ysin(\sqrt{|y|}), \ -500 < x, y < 500 \tag{5.10}
\]

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).
CHAPTER 5. PARALLEL PARTICLE SWARM OPTIMIZATION

Figure 5.1. Object function $F_2$.

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

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 1996a).

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

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:

\[
V_{i,j}^{t+1} = W_t V_{i,j}^t + C_1 r_1 (P_{i,j}^t - X_{i,j}^t) + C_2 r_2 (G_{j}^t - X_{i,j}^t) \quad (5.11)
\]

\[
X_{i,j}^{t+1} = X_{i,j}^t + V_{i,j}^{t+1} \quad (5.12)
\]

\[
f(G^t) \leq f(G_{j}^t) \quad (5.13)
\]

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 \(j\)th group, \(X_{i,j}^t\) is the \(i\)th particle position in the \(j\)th group at the \(t\)th iteration, \(V_{i,j}^t\) is the velocity of the \(i\)th particle in the \(j\)th group at the \(t\)th iteration, \(G_{j}^t\) is the best position among all particles of the \(j\)th group from the first iteration to the \(t\)th iteration and \(G^t\) is the best position among all particles in all groups from the first iteration to the \(t\)th 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.
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
three communication strategies is as follows:

1. **Initialization**: Generate $N_j$ particles $X_{i,j}^t$ for the $j$th 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 $j$th 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 $j$th 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:

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

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

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

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

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

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>
<thead>
<tr>
<th>Function</th>
<th>Asymmetric Initialization Range</th>
<th>$V_{\text{max}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f_1$</td>
<td>$15 \leq x_i \leq 30$</td>
<td>100</td>
</tr>
<tr>
<td>$f_2$</td>
<td>$2.56 \leq x_i \leq 5.12$</td>
<td>10</td>
</tr>
<tr>
<td>$f_3$</td>
<td>$300 \leq x_i \leq 600$</td>
<td>600</td>
</tr>
</tbody>
</table>

To ensure a fair comparison, the number of groups × 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 $1 \times 160$. 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. $8 \times 20$), 4 groups with 40 particles in each group (i.e. $4 \times 40$) and 2 groups with 80 particles in each group (i.e. $2 \times 80$), 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

<table>
<thead>
<tr>
<th>Percentage of Migration</th>
<th>cost of $f_1(X)$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PSO</td>
</tr>
<tr>
<td>None</td>
<td>108.74</td>
</tr>
<tr>
<td>25%</td>
<td>65.38</td>
</tr>
<tr>
<td>50%</td>
<td>75.99</td>
</tr>
<tr>
<td>75%</td>
<td>61.19</td>
</tr>
<tr>
<td>100%</td>
<td>68.44</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Percentage of Migration</th>
<th>cost of $f_2(X)$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PSO</td>
</tr>
<tr>
<td>None</td>
<td>24.54</td>
</tr>
<tr>
<td>25%</td>
<td>16.88</td>
</tr>
<tr>
<td>50%</td>
<td>15.12</td>
</tr>
<tr>
<td>75%</td>
<td>12.88</td>
</tr>
<tr>
<td>100%</td>
<td>11.24</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Number of Migration</th>
<th>cost of $f_3(X)$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PSO</td>
</tr>
<tr>
<td>None</td>
<td>0.01191</td>
</tr>
<tr>
<td>1</td>
<td>0.01028</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Function</th>
<th>cost</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PSO</td>
<td>PPSO(4,40)</td>
<td>PPSO(8,20)</td>
</tr>
<tr>
<td>Rosenbrock</td>
<td>108.74</td>
<td>76.59</td>
<td>82.62</td>
</tr>
<tr>
<td>Rastrigin</td>
<td>24.54</td>
<td>18.63</td>
<td>19.14</td>
</tr>
<tr>
<td>Griewank</td>
<td>0.01191</td>
<td>0.01053</td>
<td>0.00989</td>
</tr>
</tbody>
</table>
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 \textit{PACS}. Experimental results based on the traveling salesman problem confirm the efficiency and effectiveness of the proposed \textit{PACS} (Chu, Roddick, Pan & Su 2003, Chu, Roddick & Pan 2004a).

6.1 History of Ant System and Ant Colony System

The Ant System algorithm (Coloni, Dorigo & Maniezzo 1991, Dorigo et al. 1996) is a cooperative population-based search algorithm inspired by the behaviour of real ants. As each ant constructs 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 \textit{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 \( \tau_0 \) 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
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 \( k \)th ant as

\[
P_k(r, s) = \begin{cases} 
\frac{[\tau(r,s)]^{a} \cdot [\eta(r,s)]^{b}}{\sum_{u \in J_k(r)} [\tau(r,u)]^{a} \cdot [\eta(r,u)]^{b}}, & \text{if } s \in J_k(r) \\
0, & \text{otherwise}
\end{cases} \tag{6.1}
\]
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 \( k \)th ant positioned on city \( r \), and \( \beta \) is a parameter which determines the relative importance of pheromone level versus distance. Select the next visited city \( s \) for the \( k \)th 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

\[
\tau(r, s) \leftarrow (1 - \alpha) \cdot \tau(r, s) + \sum_{k=1}^{m} \Delta \tau_k(r, s) \quad (6.2)
\]

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

\( \Delta \tau_k(r, s) \) is the pheromone level laid down between cities \( r \) and \( s \) by the \( k \)th ant, \( L_k \) is the length of the route visited by the \( k \)th 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
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 $\tau_0$. Set the cycle counter to be 0.

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

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

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 $\beta$ is a parameter which determines the relative importance of pheromone level versus distance.

**Step 3:** Update the pheromone level between cities as

$$
\tau(r, s) \leftarrow (1 - \rho) \cdot \tau(r, s) + \rho \cdot \Delta \tau(r, s) \quad (6.5)
$$

where $\Delta \tau(r, s) = \tau_0 = (n \cdot L_{nn})^{-1}$ 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

$$
\tau(r, s) \leftarrow (1 - \alpha) \cdot \tau(r, s) + \alpha \cdot \Delta \tau(r, s) \quad (6.6)
$$
\[ \Delta \tau(r, s) = \begin{cases} (L_{gb})^{-1}, & \text{if } (r, s) \in \text{global best route} \\ 0, & \text{otherwise} \end{cases} \]  

(6.7)

where \( L_{gb} \) is the length of the shortest route and \( \alpha \) 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 (Cohon et al. 1987). The resulting parallel genetic algorithm has been successfully applied to noise reduction of vector quantization based communication (Pan et al. 1996a).

#### 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 2004a) is proposed. The Parallel Ant Colony System (PACS) is described as follows:

**Step 1: Initialization** – Generate \( N_j \) artificial ants for the \( j \)th group, \( j = 0, 1 \ldots G - 1 \). \( N_j \) and \( G \) are the number of artificial ants for the \( j \)th 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 \( \tau_0 \). Set the cycle counter to be 0.

**Step 2: Movement** – Calculate the next visited city \( s \) for the \( i \)th ant in the \( j \)th group according to

\[
s = \arg \max_{u \in J_{i,j}(r)} [\tau_j(r,u)] \cdot [\eta(r,u)]^\beta, \quad \text{if } q \leq q_0 \quad \text{(exploitation)}
\]

visit city \( s \) with \( P_{i,j}(r,s) \), \( \text{if } q > q_0 \quad \text{(biased exploration)}
\]

\[
P_{i,j}(r,s) = \begin{cases} \frac{[\tau_j(r,s)]^{\beta} \cdot [\eta(r,s)]^{\beta}}{\sum_{u \in J_{i,j}(r)} [\tau_j(r,u)]^{\beta} \cdot [\eta(r,u)]^{\beta}}, & \text{if } s \in J_{i,j}(r) \\ 0, & \text{otherwise} \end{cases}
\]

where \( P_{i,j}(r,s) \) is the transition probability from city \( r \) to city \( s \) for the \( i \)th ant in the \( j \)th group. \( \tau_j(r,s) \) is the pheromone level between city \( r \) to city \( s \) in the \( j \)th 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 \( i \)th ant in the \( j \)th group and \( \beta \) 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

\[
\tau_j(r,s) \leftarrow (1 - \rho) \cdot \tau_j(r,s) + \rho \cdot \Delta \tau(r,s)
\]

\[
\Delta \tau(r,s) = \tau_0 = (n \cdot L_{nn})^{-1}
\]

where \( \tau_j(r,s) \) is the pheromone level between cities \( r \) and \( s \) for the ants in the \( j \)th 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

\[
\tau_j(r, s) \leftarrow (1 - \alpha) \cdot \tau_j(r, s) + \alpha \cdot \Delta \tau_j(r, s)
\]

\[
\Delta \tau_j(r, s) = \begin{cases} 
(L_j)^{-1}, & \text{if } (r, s) \in \text{best route of } j\text{th group} \\
0, & \text{otherwise}
\end{cases}
\]

where \( L_j \) is the shortest length for the ants in the \( j \)th group and \( \alpha \) is a pheromone decay parameter.

Step 6: Updating From Communication – Seven communication strategies are proposed as follows:

- **Strategy 1:** As shown in Figure 6.4, update the pheromone level between cities for each group for every \( R_1 \) cycles as

\[
\tau_j(r, s) \leftarrow \tau_j(r, s) + \lambda \cdot \Delta \tau_{best}(r, s)
\]
\[ \Delta \tau_{\text{best}}(r, s) = \begin{cases} (L_{gb})^{-1}, & \text{if } (r, s) \in \text{best route of all groups} \\ 0, & \text{otherwise} \end{cases} \]

where \( \lambda \) 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

\[
\tau_j(r, s) \leftarrow \tau_j(r, s) + \lambda \cdot \Delta \tau_{\text{ng}}(r, s)
\]

\[ \Delta \tau_{\text{ng}}(r, s) = \begin{cases} (L_{\text{ng}})^{-1}, & \text{if } (r, s) \in \text{best route of neighbour group} \\ 0, & \text{otherwise} \end{cases} \]

where *neighbour* is defined as being the group whose binary representation of the group number \( j \) differs by the least significant bit. \( \lambda \) is a pheromone decay parameter and \( L_{\text{ng}} \) 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

$$\tau_j(r, s) \leftarrow \tau_j(r, s) + \lambda \cdot \Delta \tau_{ng}(r, s)$$

$$\Delta \tau_{ng}(r, s) = \begin{cases} (L_{ng})^{-1} & \text{if } (r, s) \in \text{best route of neighbour group} \\ 0 & \text{otherwise} \end{cases}$$

where \textit{neighbour} is defined as being the group arranged as the ring structure. $\lambda$ 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

$$\tau_j(r, s) \leftarrow \tau_j(r, s) + \lambda \cdot \Delta \tau_{ng}(r, s)$$

$$\Delta \tau_{ng}(r, s) = \begin{cases} (L_{ng})^{-1} & \text{if } (r, s) \in \text{best route of neighbour group} \\ 0 & \text{otherwise} \end{cases}$$

where \textit{neighbour} is defined as being those groups where the binary representation of the group number $j$ differs by one bit. $\lambda$ 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 $1 \times 80$. 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. $4 \times 20$) and 8 groups with 10 ants in each group (i.e. $8 \times 10$), 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.](image)

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
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 & Coloni 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 ∼ 7) obtained in comparison with AS and ACS for EIL101 data set on TSP problem

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>730</td>
<td>683</td>
<td>657</td>
<td>655</td>
<td>648</td>
<td>653</td>
<td>649</td>
<td>645</td>
<td>654</td>
</tr>
<tr>
<td>2</td>
<td>730</td>
<td>680</td>
<td>657</td>
<td>655</td>
<td>650</td>
<td>647</td>
<td>643</td>
<td>648</td>
<td>647</td>
</tr>
<tr>
<td>3</td>
<td>731</td>
<td>681</td>
<td>644</td>
<td>646</td>
<td>655</td>
<td>655</td>
<td>641</td>
<td>646</td>
<td>653</td>
</tr>
<tr>
<td>4</td>
<td>720</td>
<td>678</td>
<td>645</td>
<td>648</td>
<td>651</td>
<td>654</td>
<td>647</td>
<td>647</td>
<td>653</td>
</tr>
<tr>
<td>5</td>
<td>727</td>
<td>676</td>
<td>641</td>
<td>643</td>
<td>648</td>
<td>663</td>
<td>648</td>
<td>656</td>
<td>651</td>
</tr>
<tr>
<td>6</td>
<td>727</td>
<td>673</td>
<td>656</td>
<td>655</td>
<td>648</td>
<td>644</td>
<td>645</td>
<td>655</td>
<td>648</td>
</tr>
<tr>
<td>7</td>
<td>698</td>
<td>675</td>
<td>642</td>
<td>644</td>
<td>649</td>
<td>658</td>
<td>646</td>
<td>648</td>
<td>651</td>
</tr>
<tr>
<td>8</td>
<td>726</td>
<td>679</td>
<td>646</td>
<td>651</td>
<td>653</td>
<td>662</td>
<td>658</td>
<td>645</td>
<td>647</td>
</tr>
<tr>
<td>9</td>
<td>721</td>
<td>672</td>
<td>645</td>
<td>646</td>
<td>651</td>
<td>656</td>
<td>652</td>
<td>642</td>
<td>649</td>
</tr>
<tr>
<td>10</td>
<td>718</td>
<td>685</td>
<td>643</td>
<td>651</td>
<td>654</td>
<td>647</td>
<td>645</td>
<td>652</td>
<td>651</td>
</tr>
<tr>
<td>Average</td>
<td>723</td>
<td>678</td>
<td>648</td>
<td>649</td>
<td>651</td>
<td>654</td>
<td>648</td>
<td>648</td>
<td>650</td>
</tr>
</tbody>
</table>

The performance is measured in terms of solution quality, with lower values indicating better performance.
Table 6.2: The performance of ACS with communication strategies (strategy 1 ∼ 7) obtained in comparison with AS and ACS for ST70 data set on TSP problem

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1.80</td>
<td>1.80</td>
<td>4.20</td>
<td>8.10</td>
<td>4.20</td>
<td>8.10</td>
<td>4.20</td>
<td>8.10</td>
<td>4.20</td>
</tr>
<tr>
<td>1</td>
<td>734</td>
<td>701</td>
<td>679</td>
<td>680</td>
<td>683</td>
<td>684</td>
<td>678</td>
<td>683</td>
<td>681</td>
</tr>
<tr>
<td>2</td>
<td>721</td>
<td>700</td>
<td>681</td>
<td>681</td>
<td>677</td>
<td>686</td>
<td>688</td>
<td>681</td>
<td>677</td>
</tr>
<tr>
<td>3</td>
<td>722</td>
<td>700</td>
<td>681</td>
<td>682</td>
<td>678</td>
<td>683</td>
<td>678</td>
<td>681</td>
<td>678</td>
</tr>
<tr>
<td>4</td>
<td>717</td>
<td>701</td>
<td>677</td>
<td>688</td>
<td>682</td>
<td>687</td>
<td>678</td>
<td>682</td>
<td>679</td>
</tr>
<tr>
<td>5</td>
<td>721</td>
<td>703</td>
<td>678</td>
<td>684</td>
<td>678</td>
<td>686</td>
<td>678</td>
<td>681</td>
<td>678</td>
</tr>
<tr>
<td>6</td>
<td>713</td>
<td>702</td>
<td>691</td>
<td>690</td>
<td>694</td>
<td>683</td>
<td>678</td>
<td>682</td>
<td>678</td>
</tr>
<tr>
<td>7</td>
<td>714</td>
<td>700</td>
<td>682</td>
<td>683</td>
<td>678</td>
<td>683</td>
<td>677</td>
<td>683</td>
<td>681</td>
</tr>
<tr>
<td>8</td>
<td>730</td>
<td>701</td>
<td>677</td>
<td>682</td>
<td>677</td>
<td>679</td>
<td>677</td>
<td>682</td>
<td>681</td>
</tr>
<tr>
<td>9</td>
<td>730</td>
<td>696</td>
<td>677</td>
<td>681</td>
<td>679</td>
<td>685</td>
<td>678</td>
<td>678</td>
<td>677</td>
</tr>
<tr>
<td>10</td>
<td>736</td>
<td>699</td>
<td>678</td>
<td>693</td>
<td>682</td>
<td>688</td>
<td>678</td>
<td>681</td>
<td>681</td>
</tr>
<tr>
<td>Average</td>
<td>724</td>
<td>700</td>
<td>680</td>
<td>684</td>
<td>681</td>
<td>684</td>
<td>679</td>
<td>681</td>
<td>679</td>
</tr>
</tbody>
</table>
Table 6.3: The performance of *ACS* with communication strategies (strategy 1 ∼ 7) obtained in comparison with *AS* and *ACS* for TSP225 data set on TSP problem

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4587</td>
<td>4145</td>
<td>3907</td>
<td>3933</td>
<td>3913</td>
<td>3879</td>
<td>3885</td>
<td>3903</td>
<td>3949</td>
</tr>
<tr>
<td>2</td>
<td>4492</td>
<td>4215</td>
<td>3903</td>
<td>3883</td>
<td>3879</td>
<td>3879</td>
<td>3883</td>
<td>3877</td>
<td>3955</td>
</tr>
<tr>
<td>3</td>
<td>4454</td>
<td>4149</td>
<td>3888</td>
<td>3926</td>
<td>3900</td>
<td>3890</td>
<td>3889</td>
<td>3896</td>
<td>3953</td>
</tr>
<tr>
<td>4</td>
<td>4609</td>
<td>4160</td>
<td>3892</td>
<td>3886</td>
<td>3908</td>
<td>3952</td>
<td>3889</td>
<td>3885</td>
<td>3895</td>
</tr>
<tr>
<td>5</td>
<td>4538</td>
<td>4163</td>
<td>3881</td>
<td>3869</td>
<td>3888</td>
<td>3898</td>
<td>3879</td>
<td>3885</td>
<td>3879</td>
</tr>
<tr>
<td>6</td>
<td>4483</td>
<td>4146</td>
<td>3942</td>
<td>3915</td>
<td>3916</td>
<td>3978</td>
<td>3892</td>
<td>3901</td>
<td>3961</td>
</tr>
<tr>
<td>7</td>
<td>4555</td>
<td>4149</td>
<td>3904</td>
<td>3911</td>
<td>3876</td>
<td>3939</td>
<td>3881</td>
<td>3891</td>
<td>3881</td>
</tr>
<tr>
<td>8</td>
<td>4491</td>
<td>4148</td>
<td>3950</td>
<td>3900</td>
<td>3912</td>
<td>3925</td>
<td>3950</td>
<td>3889</td>
<td>3890</td>
</tr>
<tr>
<td>9</td>
<td>4500</td>
<td>4108</td>
<td>3903</td>
<td>3916</td>
<td>3903</td>
<td>3894</td>
<td>3889</td>
<td>3887</td>
<td>3896</td>
</tr>
<tr>
<td>10</td>
<td>4521</td>
<td>4161</td>
<td>3877</td>
<td>3915</td>
<td>3875</td>
<td>3911</td>
<td>3877</td>
<td>3896</td>
<td>3876</td>
</tr>
<tr>
<td>Average</td>
<td>4523</td>
<td>4154</td>
<td>3905</td>
<td>3905</td>
<td>3897</td>
<td>3920</td>
<td>3891</td>
<td>3889</td>
<td>3909</td>
</tr>
</tbody>
</table>
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 constraints 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.
CHAPTER 6. PARALLEL AND CONSTRAINED ANT COLONY OPTIMIZATIONS 196

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. 1996b) 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

\[ D_q(O, X_m) = (O - X_m)^t W^{-1} (O - X_m) \]  \hspace{1cm} (6.8)

where \((O - X_m)\) is error column vector and \(W\) is the covariance matrix given as

\[ W = \frac{1}{T} \sum_{i=1}^{T} (X_i - \bar{X})(X_i - \bar{X})^t \]  \hspace{1cm} (6.9)

where \(\bar{X}\) is the mean of \(X_i, i = 1, 2, \cdots, T\) defined as

\[ \bar{X} = \frac{1}{T} \sum_{i=1}^{T} X_i \]  \hspace{1cm} (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.

\[
L^k_{ts} = \text{Avg}L^k_{path} + \text{StDev}L^k_{path}
\]

where \( \text{Avg}L^k_{path} \) and \( \text{StDev}L^k_{path} \) are the average of the distance and the standard deviation for the route of the visited objects by the \( k \)th ant expressed as

\[
\text{Avg}L^k_{path} = \frac{\sum L^k_{ij}}{E}, \quad \text{if } (X_i, X_j) \text{ path visited by the } k \text{th ant}
\]

\[
\text{StDev}L^k_{path} = \sqrt{\frac{\sum (L^k_{ij} - \text{Avg}L^k_{path})^2}{E}},
\]

\[
\text{if } (X_i, X_j) \text{ path visited by the } k \text{th ant}
\]

where \( E \) is the number of paths visited by the \( k \)th ant. We may roughly consider object \( X_i \) and object \( X_j \) are located in different clusters if \( L^k_{ij} > L^k_{ts} \). 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

\[
P_k(r, s) = \begin{cases} 
\frac{[\tau(r,s)]^{-\beta}|D_q(r,s)|^{-\gamma}|SKNND(s)|^{-\gamma}}{\sum_{u \in J^N_k(r)}[\tau(r,u)]^{-\beta}|D_q(r,u)|^{-\gamma}|SKNND(u)|^{-\gamma}}, & \text{if } s \in J^N_k(r) \\
0, & \text{otherwise}
\end{cases}
\]

(6.14)
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. $\beta$ and $\gamma$ 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. $\beta$ is set to 2 and $\gamma$ 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 $\tau_{ij}$ between any two objects $X_i$ and $X_j$ is set to be a small positive constant $\tau_0$.

**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.
CHAPTER 6. PARALLEL AND CONSTRAINED ANT COLONY OPTIMIZATIONS 200

Figure 6.13. Shrinking search route using Equation 6.14

Step 3: Pheromone Update

Update the pheromone level between objects as

\[ \tau_{ij}(t + 1) = (1 - \alpha)\tau_{ij}(t) + \Delta\tau_{ij}(t + 1) \]  

\[ \Delta\tau_{ij}(t + 1) = \sum_{k=1}^{T} \Delta\tau_{ij}^k(t + 1) \]  

\[ \Delta\tau_{ij}^k(t + 1) = \begin{cases} \frac{Q}{L_k}, & \text{if } ((i, j) \in \text{route done by ant } k \text{ and } L_{ij}^k < L_{ts}^k) \\ 0, & \text{otherwise} \end{cases} \]  

where \( \tau_{ij} \) is the pheromone level between object \( X_i \) and object \( X_j \), \( T \) is the total number of clustering objects, \( \alpha \) 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 \( k \)th ant.

Step 4: Consolidation

Calculate the average pheromone level on the route for all objects as

\[ Avg\tau = \frac{\sum_{i,j \in E} \tau_{ij}}{E} \]  

where \( E \) is the number of paths visited by the \( k \)th ant. Disconnect the path between two objects if the pheromone level between these two objects
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 $\gamma$ 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.

$\gamma$ 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
into just two clusters if $\gamma$ is set to 1. By setting $\gamma$ to 5, the Four-Clusters data set can be correctly partitioned as shown in Figure 6.17. We found that $\gamma$ 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 $\epsilon$ 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
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
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 \textit{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 \textit{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. \textit{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 \textit{CACO} 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 (MCMS) and Incremental Multi-Centroid, Multi-Run Sampling Scheme (IMCMS) are proposed. These two sampling schemes can also combine with the CLASA and the other $k$-medoids algorithms. Comparing with the CLARANS algorithm, the IMCMS 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 IMCMS 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-
olution 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) 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


Journal paper


APPENDIX A. PUBLICATIONS ARISING FROM THIS THESIS


Conference paper


APPENDIX A. PUBLICATIONS ARISING FROM THIS THESIS


Bibliography


Guha, S., Rastogi, R. & Shim, K. (1999), ROCK: A robust clustering algorithm for cate-
gorical attributes, in ‘1999 International Conference Data Engineering (ICDE’99)’, Sydney, Australia, pp. 512–521.


Holland, J. (1975), *Adaptation In Natural and Artificial Systems*, University of Michigan Press.


Mobasher, B., Cooley, R. & Srivastava, J. (1999), Creating adaptive web sites through usage-based clustering of urls, in ‘Knowledge and Data Engineering Workshop’.


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. (2002), ‘Clarans: A method for clustering objects for spatial data mining’, IEEE Transactions on Knowledge and Data Engineering 14(5), 1003–1016.


