Ant Colony Optimization Based Clustering for Data Partitioning

WOO Kwan Ho

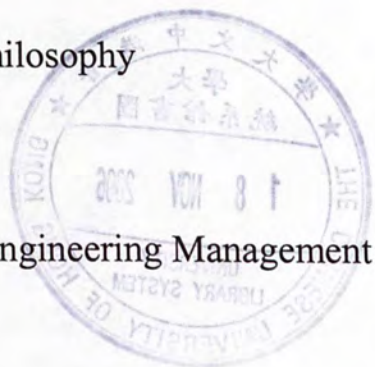
A Thesis Submitted in Partial Fulfillment

of the Requirements for the Degree of

Master of Philosophy

in

Systems Engineering and Engineering Management



August 2005


香港中文大學
大學
圖書館系統
18 NOV 2006
UNIVERSITY
LIBRARY SYSTEM
THE CHINESE UNIVERSITY OF HONG KONG

Contents

Contents ii
Abstract iv
Acknowledgements vii
List of Figures viii
List of Tables x
Chapter 1 Introduction 1
Chapter 2 Literature Reviews 7
2.1 Block Clustering 7
2.2 Clustering XML by structure 10
2.2.1 Definition of XML schematic information 10
2.2.2 Identification of XML schematic information 12
Chapter 3 Bi-Tour Ant Colony Optimization for diagonal clustering 15
3.1 Motivation 15
3.2 Framework of Bi-Tour Ant Colony Algorithm 21
3.3 Re-order of the data matrix in BTACO clustering method 27
3.3.1 Review of Ant Colony Optimization 29
3.3.2 Bi-Tour Ant Colony Optimization 36
3.4 Determination of partitioning scheme 44
3.4.1 Weighed Sum of Error (WSE) 48
3.4.2 Materialization of partitioning scheme via hypothetic matrix 50
3.4.3 Search of best-fit hypothetic matrix 52
3.4.4 Dynamic programming approach 53
3.4.5 Heuristic partitioning approach 57
3.5 Experimental Study 62
3.5.1 Data set 63
3.5.2 Study on DP Approach and HP Approach 65
3.5.3 Study on parameter settings 69
3.5.4 Comparison with GA-based & hierarchical clustering methods 81
3.6 Chapter conclusion 90
Chapter 4 Application of BTACO-based clustering in XML database system 93
4.1 Introduction 93
4.2 Overview of normalization and vertical partitioning in relational DB design 95
4.2.1 Normalization of relational models in database design 95

4.2.2 Vertical partitioning in database design 98
4.3 Clustering XML documents 100
4.4 Proposed approach using BTACO-based clustering 103
4.4.1 Clustering XML documents by structure 103
4.4.2 Clustering XML documents by user transaction patterns 109
4.4.3 Implementation of Query Manager for our experimental study 114
4.5 Experimental Study 118
4.5.1 Experimental Study on the clustering by structure 118
4.5.2 Experimental Study on the clustering by user access patterns 133
4.6 Chapter conclusion 141
Chapter 5 Conclusions 143
5.1 Contributions 144
5.2 Future works 146
Bibliography 148
Appendix I 156
Appendix II 168
Index tables for Profile A 168
Index tables for Profile B 171
Appendix III 174

Abstract

Clustering refers to a process of grouping similar objects together while separating the dissimilar objects from the same group. Traditional clustering method usually clusters the set of objects only for analysis. Block clustering is a specific clustering method which clusters the sets of objects and their associated attributes (descriptors) together by clustering both sets of data simultaneously. Hence, the set of attributes which is most relevant to the cluster of objects can be revealed. In this paper, we propose using an Ant Colony Optimization based algorithm, Bi-tour Ant Colony Optimization (BTACO) to perform the block clustering.

Consider the data matrix representing the sets of objects and their associated attributes. BTACO clustering method works by re-arranging the data matrix into a diagonal block structure and hence identifying the block clusters along the diagonal line of the re-arranged data matrix. In the computational study, we show that our proposed clustering method is well performed compared with traditional clustering methods.

In this work, we have also proposed the use of the BTACO clustering method to improve the performance of an XML database system by clustering a collection of XML documents. There are two different ways of clustering the collections: clustering by structure and clustering by user transaction pattern. The former clustering method can identify the schemas of XML documents so as to improve the query efficiency by means of analyzing the schemas. On the other hand, the latter

clustering method can partition the collection of XML documents based on the user transaction pattern so that irrelevant data may be avoided in query processing. In the experimental study, we can see that BTACO-based clustering can identify the schemas with higher accuracy compared with the other methods. Also, it can partition the collection to improve the query process significantly.

論文概述

資料分群是一個把叢集裏的資料分門別類，讓相似的資料聚集成群的程序。方塊資料分群則是一種不但可以把資料分門別類、連帶其屬性也能進行分類的資料分群方法。對於被方塊資料分群處理過的資料，我們非但能獲悉資料之間的相似關係；對於其屬性在資料之間相似度的影響，我們也能獲得理解。在本論文裏，我們將探討利用雙行螞蟻系統來進行方塊資料分群。

雙行螞蟻系統是一個經改良的螞蟻系統。對於其分群的能力，我們分別利用傳統方法和雙行螞蟻系統來進行對照實驗。根據實驗結果，雙行螞蟻系統的表現比其他的方法較爲出色：它對分群的準確度較傳統方法爲高。

在本論文裏，我們也會探討利用雙行螞蟻系統來改良ＸＭＬ的查找能力。我們將提出兩大方針來改良查找能力。第一，我們將利用雙行螞蟻系統來對ＸＭＬ文檔集進行資料分群，從而得到它們的設計構圖集。查找能力將因得到設計構圖集而得到提升。第二，我們透過利用雙行螞蟻系統來分析用戶的使用情況，從而設計出一個合適的分群數據庫並使其查找效率得到提升。根據實驗結果，雙行螞蟻系統非但能找出合適的設計構圖集，而且還能設計出一個較高效能的分群數據庫。

Acknowledgements

During the writing of this thesis, I received many helps and supports from the persons whom I would like to thank.

First of all, I would like to express my sincere gratitude to my supervisor, Professor C.H. Cheng, for his guidance and support in my research. He gave me many valuable suggestions and directions. This thesis would not be completed without his help.

Secondly, I wish to thank my internal reviewers, Professor Wai Lam and Professor Jeffrey Xu Yu, for their careful reading of my thesis and their comments.

I would also like to thank the staff of the Department of Systems Engineering and Engineering Management for their help. Thanks to Keith, Camel, Mei Choi, Ray, Walter, Michael, Lam Chai, Gabriel, Terrence, and other school fellows who make my study joyful and unforgettable.

Finally, I would like to express my deep gratitude to my family for all their love, understand and care. Without their support and encouragement, this thesis will never be completed.

List of Figures

Figure 3-1Transaction / Attribute matrix (1) 16
Figure 3-2 Re-arranged transaction / attribute matrix (2) 16
Figure 3-3 Another transaction / attribute matrix (3) 17
Figure 3-4 Matrix (4) 19
Figure 3-5 Matrix (5) 19
Figure 3-6 matrix with the access swapped (6) 20
Figure 3-7 The re-arranged data matrix generated by GA-based algorithm 22
Figure 3-8 Framework of BTACO 26
Figure 3-9 Life-cycle of each ant in BTACO 26
Figure 3-10 Pseudo code for Ant System 29
Figure 3-11 Pseudo code for Ant Colony System 29
Figure 3-12 Median Centering Technique 40
Figure 3-13 Relationship between ant algorithm, evaluation scheme and exploration scheme 47
Figure 3-14 How WSE used to evaluate the partitioning scheme 51
Figure 3-15 Definition of sub-block notation for easier illustration 53
Figure 3-16 Pseudo code for dynamic programming approach 54
Figure 3-17 Illustration for efficient calculation of WSE 56
Figure 3-18 Criterion for heuristic binary partitioning approach 57
Figure 3-19 Pseudo code for heuristic partitioning approach 60
Figure 3-20 Parameters for matrix generator 63
Figure 3-21 Comparison between DP approach and HP approach without learning 67
Figure 3-22 Data matrix 1 – Adapted from Navathe et al example 70
Figure 3-23 Data matrix 2 70
Figure 3-24 data matrix 3 71
Figure 3-25 Effect of delta(δ) and beta(β) on the solution found in **data matrix 1** without learning effect 72
Figure 3-26 Effect of delta(δ) and beta(β) on the solution found in **data matrix 1** with learning effect 72
Figure 3-27 Effect of delta(δ) and beta(β) on the solution found in **data matrix 2** without learning effect 72
Figure 3-28 Effect of delta(δ) and beta(β) on the solution found in **data**

matrix 2 with learning effect 72
Figure 3-29 Effect of delta(δ) and beta(β) on the solution found in **data matrix 3** without learning effect 72
Figure 3-30 Effect of delta(δ) and beta(β) on the solution found in **data matrix 3** with learning effect 72
Figure 3-31 Improvement curve for β = 1 in **data matrix 2** 73
Figure 3-32 Improvement curve for β = 3 in **data matrix 2** 73
Figure 3-33 Improvement curve for β = 5 in **data matrix 2** 73
Figure 3-34 Improvement curve for β = 1 in **data matrix 3** 74
Figure 3-35 Improvement curve for β = 3 in **data matrix 3** 74
Figure 3-36 Improvement curve for β = 5 in **data matrix 3** 74
Figure 3-37 Effect of learning parameters on the solution found in data matrix 2 79
Figure 3-38 Effect of learning parameters on the solution found in data matrix 3 80
Figure 3-39 Re-arranged **data matrix 1** by using BTACO 82
Figure 3-40 Re-arranged **data matrix 1** by using GA 82
Figure 3-41 Re-arranged **data matrix 1** by using hierarchical clustering... 82
Figure 3-42 Re-arranged **data matrix 2** by BTACO 83
Figure 3-43 Re-arranged **data matrix 2** by GA 83
Figure 3-44 Re-arranged **data matrix 2** by hierarchical clustering 83
Figure 3-45 Re-arranged **data matrix 3** by BTACO 84
Figure 3-46 Re-arranged **data matrix 3** by GA 84
Figure 3-47 Re-arranged **data matrix 3** by hierarchical clustering 84
Figure 4-1 Pseudo code for the mechanism of choosing (k-1) cutting position 108
Figure 4-2 Framework of the database system for the partitioned XML collection 116
Figure 4-3 Matrix for access profile A 134
Figure 4-4 Matrix for access profile B 134
Figure 4-5 clustering result for matrix in Figure 4-2 135
Figure 4-6 clustering result for matrix in Figure 4-3 135

List of Tables

Table 3-1 Summary of the data matrices for parameter settings..... 71
Table 3-2 Performance comparisons among three different clustering methods..... 87
Table 4-1 Information about the ACM SIGMOD record in xml version... 122
Table 4-2 Table shows the input parameter setting for each synthetic collection..... 123
Table 4-3 Table shows the number of documents, the number of unique bitwise structure as well as the number of edges identified for each cluster in the synthetic collection..... 123
Table 4-4 Parameter setting for the modified BATCO-based clustering. .. 125
Table 4-5 Comparison of Hubert indices for three different methods 127
Table 4-6 Computation time for each process in clustering using edit-tree distance (in sec)..... 128
Table 4-7 Computation time for each process in using S-GRACE method (in sec)..... 128
Table 4-8 Computation time for each process using BTACO clustering method (in sec)..... 128
Table 4-9 Comparison of total computation time of three different methods (in sec)..... 129
Table 4-10 Representation of the XQuery Expression in the access profile 135
Table 4-11 Set of queries for studying the efficiency of the partitioning scheme..... 137
Table 4-12 The computation time for the set of queries in different collection (in ms) 138
Table 4-13 The improvement ratio for the set of queries in different collection (in ms) 138
Table 4-14 The weighted time for evaluation 140

Chapter 1 Introduction

Clustering is an unsupervised classification of objects into groups. It is a process of grouping similar objects together while separating the dissimilar objects from the same group. As a classical approach in data mining, cluster analysis has been addressed in different contexts and by researchers in different disciplines. The applications of cluster analysis varies from imaging processing, data mining to vertical partitioning problem in RDBMS[55, 63, 87, 88, 103]. Despite of its wide usefulness, the nature of clustering is itself being a combinatorial problem which is difficult to solve. Besides, the transformation of a clustering method from one context to another is not straightforward, that means it's difficult to develop a generic method which can solve problems in different domains.

In cluster analysis, objects are usually represented by a vector of measurement, or a point in multidimensional space. The determination of the groups (cluster) could be done by the distance measurement over the various dimensions in the dataset. The common distance measurement includes the Euclidean distance, Manhattan (City-block) distance, cosine function, etc.

Bi-clustering (or called Block-clustering), which deals with a two-dimensional dataset, is on the focus of this work. Bi-clustering means the simultaneous clustering of both row and column sets in a data matrix[86]. The bi-clustering method is first proposed by J.A. Hartigan in 1972[54]. It aims at clustering both row and column variables simultaneously and so there would be a direct interpretation of the clusters

in the data matrix. This is why the method is also called "Direct Clustering". The clustering process is accomplished by the subsequent division of the data matrix into sub-matrices. The division is in favour when the total square sum of error after the division is smaller than the one before the division. The clustering process is finished when there is no further division of the data matrix which can result in smaller total square sum of error. By this way, the relationship between the row and column variables could be directly interpreted by the sub-blocks identified in the data matrix.

In our work, we proposed a clustering method called Bi-Tour Ant Colony Optimization (BTACO) clustering method for the block clustering. BTACO is specific to be a non-overlapping partitioning method which is different from the ones [24, 100] for overlapping partitioning. The BTACO clustering method clusters the data matrix by first trying to re-arrange the data matrix into a diagonal block structure. Then based on the re-arranged data matrix, the clustering method clusters the structure to form sub-blocks along the diagonal line of the data matrix and evaluates the goodness-of-fit of the partitioning scheme found. Since BTACO clustering method is a heuristic search in nature, it will search for the better solution iteratively through the direction found by the previous solution and hence it can get better solution with more chance. Since the diagonal structure have to be formed in the re-arranged data matrix, the clustering method is also called as diagonal clustering [104].

Bi-Tour Ant Colony Optimization (BTACO) is in fact the modified version of the Ant Colony Optimization (ACO) algorithm. In our work, we have modified and

introduced some new ideas so that the block clustering could be performed well. First, the first appeared version of ACO algorithm is used for dealing with the traveling salesman problem (TSP) which is a problem with one set of data only. In the block clustering, we have to deal with two sets of data, say the row data and the column data. As a result, we have introduced using two tours for the partitioning problem. As the clustering method will re-arrange the data matrix into a diagonal block structure, we have introduced the median centering technique to facilitate the formation of diagonal block structure. We have used a control parameter based on the median centering so as to re-arrange the data matrix with a desirable diagonal block structure and the similar rows and columns are allocated together.

The re-arrangement of the data matrix in fact gives no information about the partitioning scheme. Hence, we have to provide a partitioning method on the re-arranged data matrix as well as the goodness-of-fit measurement for the partitioning scheme found. The traditional clustering method measures the similarity among the sets of objects and hence the validation of the partitioning scheme by mere use of distance measure[23, 55, 76]. However, the mere use of distance measure is not suitable in the block clustering as two-dimensional data is being considered. To evaluate the goodness-of-fit of the partitioning scheme, we have introduced the use of Weighted Sum of Error (WSE) for the measurement which is analogous to the evaluation of the observed distribution of data with the hypothetic distribution in the statistical study. In our proposed clustering method, we will form a hypothetic data matrix which is the representation of the partitioning scheme. We will compare the hypothetic data matrix with the observed data matrix (i.e. the re-arranged data matrix) and evaluates the discrepancy among the elements of the

data matrices by means of weighted sum of error (WSE). The higher the value of WSE, the larger the discrepancy among the elements in the data matrices and hence the particular partitioning scheme is not a suitable choice. Based on the evaluation measurement, the partitioning method will try to find out the suitable partitioning scheme by identifying the hypothetic data matrix which results in the minimized WSE. The sub-blocks in the hypothetic data matrix is in fact the partitioning scheme.

By means of the formation of a diagonal block structure in data matrix, we have introduced two approaches for finding the partitioning scheme, they are the dynamic programming approach and the heuristic partitioning approach. The two approaches differ from the effectiveness and the accuracy in finding out the partitioning scheme. The dynamic programming approach can find out a better partitioning scheme but it suffers from the quadratic time complexity. On the other hand, the heuristic partitioning approach highly depends on a well-formed diagonal block structure; the partitioning scheme is not good when the re-arranged data matrix is not in a good diagonal block structure. However, the heuristic partitioning scheme can find out a partitioning scheme in linear time.

In our work, we also proposed the use of BTACO clustering method in clustering the collection of XML documents for improving the efficiency of XML database system. There are two ways of clustering XML documents, they are by structure and by user transaction pattern. There are literatures about clustering XML documents by structure [78, 89], in which such a clustering is used for identifying the schemas of the XML documents. The query engine can make use of the schemas to search only

the relevant information for the query result. To facilitate the use of BTACO clustering method in identifying the schemas, we have modified BTACO clustering method so that it can support classifying data into k-clusters. The modified clustering method can perform well in identifying the schemas of the XML documents. On the other hand, we proposed clustering the user transaction profile for the particular collection of XML documents. Based on the user transaction pattern, the collection of the XML documents is clustered into different groups. During the query processing, the query engine will only search the relevant partition of the collections and hence the irrelevant data may be avoided. As a result, the database could perform in a more efficient way.

In our work, we have studied the performance of BTACO clustering method in block clustering and the performance in clustering XML documents. We have used the vertical partitioning problem for illustrating the performance of BTACO clustering method in block clustering. In the computational study, we have found that the BTACO out-performs the other two clustering methods, the hierarchical clustering with average distance measure and the genetic algorithm (GA) based clustering method. Our proposed clustering method could get a smaller Weighted Sum of Error (WSE) which indicated that the partitioning scheme found is better. Also, we have tested on different parameter settings and concluded that the best partitioning scheme can be formed by a well-balance between the exploration and exploitation of the solution found by the iterative searching. Also, the median centering technique provides a good direction for the clustering method to form a diagonal block structure of the data matrix. Furthermore, the heuristic partitioning approach works better than the dynamic programming approach in overall as it

could find out the partitioning scheme in a much efficient way. The scalability of using heuristic partitioning approach would be much larger than that of using the dynamic programming approach. In fact, preliminary study have been done in [104]. However, the work is lack of investigation on the effect of parameter setting. In this work, a thorough investigation is given.

In the study of clustering XML documents, we have found that our proposed clustering method can identify the schemas of the XML documents better by having a higher accuracy in identifying the groups of XML documents with similar structure. In the computational study, we test the performance of clustering user profile by using the XML version of ACM SIGMOD record for the experiment. In the experiment, we can see that the query which is in the user profile can find the result faster when the collection of XML documents is partitioned based on the user profile. Also, we can see that the performance of the database is optimized only when certain partitioning scheme is designed for the user profile found. The partitioning scheme cannot be set arbitrary.

The rest of the paper is organized as follows. The coming chapter gives a brief literature review about the block clustering methods. In chapter 3, the Bi-Tour Ant Colony Optimization (BTACO) clustering method together with its computational study is discussed. In chapter 4, we will discuss about clustering XML documents by structure and by user profile. The computational study about the performance of BTACO clustering method in clustering XML documents is also shown. Finally, we will conclude the clustering method and its application in clustering XML documents as well as the future work in chapter 5.

Chapter 2 Literature Reviews

In the followings, we will first discuss the literatures about the block clustering which is related to the development of BTACO clustering method. Then we will discuss about the clustering of XML documents by structure which is related to the use of BTACO in clustering XML documents.

2.1 Block Clustering

We discuss about different block clustering methods found in the literatures. In the followings, we can see that although all the methods work by simultaneous clustering of rows and columns of a specific data matrix, they differ from the techniques used and contexts to be solved.

Clustering is the process of organizing the similar objects together, so that the members within groups are similar while they are dissimilar between groups in some way [55, 63]. On the other hand, block clustering is a specific field of clustering analysis in which a set of objects (rows, observation) and a set of variables (columns, attributes, cases) are considered together [37, 54]. Considering a data matrix M with a set of objects and a set of variables, the block clustering methods consist of two processes. First, it makes permutations of rows and columns of the data matrix M to draw a correspondence structure, a re-arranged matrix. Then, based on the desirable permutation order, the re-arranged matrix is clustered into a set of sub-blocks for analysis [37, 49, 54].

Hartigan [54] proposed a model which could cluster the cases and variables simultaneously. As the direct interpretation of the clusters could be seen on the data matrix, the block clustering was named as direct clustering in his work. In his proposed method, the orders of row and column are first permuted to form a re-arranged data matrix in which larger contiguous sub-blocks could be identified, in which the order is based on ascendant order of the column (row) sum. Based on the permuted row and column orders, row and column splits are applied to the re-arranged data matrix which results in the reduction of the variance within the sub-blocks exist. There are two types of split, the fixed and free splits, which perform the division on the whole set of column (row) and perform division on the predetermined set respectively. In the algorithm, the process of splitting continues until there is no reduction of the variance by division further. Duffy and Quiroz [37] showed that the block clustering could be permutation-based rather than Guassian distribution-based as proposed by Hartigan [54].

With the rise of mixture model approach in cluster analysis, Govaert and Nadif [49] proposed the notion of block mixture model for the block clustering. Mixture model approach is an approach in which the algorithm models distribution to the data, which is assumed to be independent and identical distributed (i.i.d). Guassian mixture model is common for the clustering analysis. For the block mixture model, given the appropriate probability density function for each element in the data matrix M, the estimation of the parameters of the mixture model could be done by the optimization of their associated log-likelihood functions using the EM algorithm.

McCormick, Schweizer, and White [82] proposed a "bond energy" approach to

render the given data matrix into a pattern such that underlying relationship among the data could be interpreted by inspection. There have been many literatures about the bond energy algorithm for the data analysis, such as performing taxonomy (clustering) in production research, imaging and related engineering problems. [6, 58, 62, 71, 80, 96]

Since the approach could be done by solving the associated traveling salesman problem (TSP) [76], Cheng et al. [22, 23] proposed using the genetic algorithm to solve the associated traveling salesman problem for simultaneous clustering of row and column in the data matrix. Kusiak and Chow [73] proposed an iterative algorithm for cluster identification. However, the algorithm could not work properly with the noise data. Based on Kusiak and Chow's cluster identification algorithm, Cheng [21] proposed a branch and bound approach to eliminate the columns in the matrix which could avoid the formation of cluster. The resulting data matrix could have a sets of cluster with some columns are eliminated.

For studying the gene expression data, Cheng et al [24]introduced the mean squared residue score as a measure for the coherence of the genes and conditions in a DNA micro-array. The sub-matrix is called a δ-cluster if the score $\leq \delta$ for some $\delta > 0$. The sub-matrix could be resulted by performing insertion and deletion of row and column in the sub-matrix iteratively until a desired sub-matrix is formed. Yang et al [105] improved the algorithm by using a move-based algorithm. However, it suffers from outlier problem and requires the number of clusters as an input parameter. Wang et al [100] proposed the use of δ–pairwise-cluster which is a sub-matrix in which it contains the maximum dimension set of column and row with the difference

between the largest and smallest values is below δ.

Conceptual clustering methods are used to group the unlabelled objects into classes. The difference of conceptual clustering methods from the other methods is that both extensional description (i.e. the set of objects in the classes) and intensional (i.e. the set of conceptual information) description are considered together [84]. This problem definition is similar to the problem to be solved by block clustering. For the conceptual clustering methods, the clustering result can be in the form of conceptual lattice, which is a Hesse diagram-like graph to show the relationship between the sets of objects and the sets of descriptors (attributes) [16, 31]. There have been many works related to incremental hierarchical conceptual clustering[42, 52, 53, 74, 81]. The conceptual clustering considers only the binary relation between the sets of objects and the sets of descriptors. Hence, it is limited to deal with the matrix without any weighting.

2.2 Clustering XML by structure

In this section, we will discuss about the methods in clustering XML by structure. Since the objective of clustering XML by structure is to identify the schemas in the XML documents. As a result, we will first discuss about the definition of XML schematic information before reviewing the literatures about clustering XML by structure.

2.2.1 Definition of XML schematic information

For a relational database design, we may use the normalization techniques as

discussed for designing the schema in the database system. The normalized schemas can avoid the data redundancy and anomalies so as to provide efficient storage model for the database system[95, 102]. In fact, schematic information is also necessary for the storage of the XML documents. In the XML repositories, the schematic information could allow a query to access the relevant portions of the data only, resulting in greater efficiency. Document Type Descriptors (DTD) and the XML schema are the most common schematic information for the XML documents.

2.2.1.1 Document Type Descriptors (DTD)

Document Type Descriptors (DTD)[13, 38, 98] is mainly used to define the internal structure, content and the semantics of the XML documents. The format of the XML document may be restricted by defining the position, the occurrence, and the semantics of the information to be represented in the XML documents. For defining position, it restricts the sequence as well as nested relationship of the information in the documents. For the occurrence, it restricts the cardinality of the information (say, the element) to be present in the structure. Finally, for the semantics, it defines the legal building elements of an XML document in the nested structure. The collection of elements consists of tag, attribute, entity and character data (PCDATA and CDATA).

2.2.1.2 XML Schema

XML schema[97], another format of schematic information for XML documents, provides a means similar to the DTD. However, it provides more flexibility than the DTD. XML schema is an XML-type based format to define the XML documents, in

which it enhances the definition of the schema as follow:

- **More flexibility in type definition of the element**

In the XML schema, more flexibility is given to define the type of data to be in the element nodes, which is the primary atomic node for storing the node name, node value and attributes(if any). For example, the textual information in a node could be restricted as integer, string, data, etc. Also, enumeration may be applied so that the specific node is allowed to be one of the choices in the list of textual data.

- **More flexibility in occurrence constraints**

In DTD, the cardinality is restricted by one of the symbols "?", "*" and "+" which specify "zero or one", "zero or more" and "one or more" respectively. However, in XML schema, the cardinality is restricted by the minimum and maximum occurrence which provides a more flexibility in the format.

Besides DTD and XML schema, there are many other formats of XML schematic information, such as XDR[43], DSD[70], SOX[32] and Schematron[64]. However, DTD and XML schema are more widely known and are well support[90]. As a result, we discuss about them only.

2.2.2 Identification of XML schematic information

Although there are standard formats for defining the schematic information of the XML documents, it is not necessary for the XML documents to be embedded with the schematic information, especially for the early XML documentation which is

extracted from the web[89]. Also, even with the existence of the schematic information, the same set of information could be represented by different format and hence different schemas. As a result, we have to perform data integration in the collection of XML documents[75] for efficient data storage. Different approaches in automatic identification of schemas are necessary and have been developed. In the earlier approach, work is solely based on generalizing the homogeneous collection of XML documents. Later, the work is extended to deal with the heterogeneous collection of XML documents by defining a suitable distance measure for the structural similarity of the XML documents.

Garofalakis et al[47] proposed a method to automatically extract a DTD from a set of XML documents. It generates a DTD by the adaptation of algorithms for logic optimization and Minimum Description Length (MDL) principle so as to form the most suitable and general DTD for the set of XML documents. Data Descriptors by Example (DDbE) [12] is a Java component library which performs the extraction similarly. By the DTD extraction methods, one has to determine which XML documents should be grouped together for finding out a common general DTD. That is, the method could only be used for the homogeneous collection of XML documents.

Because of this problem, Nierman and Jagadish[89] proposed using the edit-tree distance measure to find out the groups of XML documents with similar structure before the extraction of the DTD schema. The distance is measured by finding out the cheapest sequence of edit operations which could transform the source tree to the destination tree[17, 18, 107]. By the edit-tree distance measurement, the XML

documents are expressed as an ordered labeled tree[57]. In order to find the allowable edit sequences, there are 5 edit operations: *Re-label*, *Insert*, *Delete*, *Insert Tree* and *Delete Tree* for transformation of tree. There are also two conditions to be satisfied so as to prevent from unreasonable grafting(sub-tree insertion) and pruning(sub-tree deletion). The similarity cost then is equal to the minimum cost of the sequence of allowable edit operations required to transform from the one (source) tree to the other (destination) tree. The cost of each pair of trees can be calculated by dynamic programming[107]. The main drawback of this measure is that the complexity of finding each pair of XML documents is great. The complexity is O(|A| |B|) which is too high for practical application when the collection of XML documents is large.

Lian et al [78] proposed an efficient evaluation of the similarity between XML documents as well as the clustering algorithm for the XML documents. The XML documents are represented by semi-structure graphs for the evaluation. The similarity between pair of XML documents is based on the number of common edges found in the pair of associated semi-structure graphs. The higher the number of common edges, the more the number of common element-subelement relationships between the XML documents and hence the degree of similarity should be higher. As the set of edges in the semi-structure graphs is represented by a set of bit strings, the clustering process is fast and scalable. Lian et al also suggested a clustering algorithm, called S-GRACE, which is derived from a categorical clustering algorithm ROCK[51] for identifying the groups of similar XML documents.

Chapter 3 Bi-Tour Ant Colony Optimization for diagonal clustering

3.1 Motivation

To demonstrate the concept of non-overlapping block clustering, vertical partitioning in a relational database system is considered[23]. In the vertical partitioning problem, an application work profile of the database design is provided. The access patterns of a set of transactions {T1, T2, ...} and a set of attributes of the database relation {A1, A2, ...} would form the application work profile of the database design. For design purpose, the application work profile is shown in a transaction-attribute matrix. Consider the transaction-attribute matrix in Figure 3-1. It contains five non-primary key attributes, i.e. {A1, A2, A3, A4, A5}, and four transactions, {T1, T2, T3, T4}. A "1" (or "0") entry in the matrix indicates that a transaction requires (or does not require) the use of an attribute(s) concerned. Notice that in the matrix (1) in Figure 3-1, the distribution of "1" in the matrix is completely random. Such a matrix is practically useless in database design.

	Attributes					Access
Transaction	*A1*	*A2*	*A3*	*A4*	*A5*	**Freq**
T1		1		1	1	10
T2	1		1			20
T3		1		1		30
T4	1		1			20

Figure 3-1Transaction / Attribute matrix (1)

Consider another transaction-attribute matrix in Figure 3-2. It is formed by rearranging certain transactions and attributes of matrix (1). Matrix (2) comprises a diagonal block structure. A diagonal block structure refers to the structure with most of "1" arranged along the diagonal. TC-1 is a transaction cluster which accesses AC-1, an attribute cluster; similarly, TC-2 accesses AC-2. TC-1/AC-1 and TC-2/AC-2 form two perfectly separable sub-matrices. Practically, the set of transactions accesses the set of attributes in the same sub-matrix. Attributes in a sub-matrix make up a fragment. This lays down to the objective of vertical partitioning.

		Attributes					
		AC-1		*AC-2*			Access
Transaction		*A1*	*A3*	*A2*	*A4*	*A5*	Freq
TC-1 [	*T2*	1	1				20
	T4	1	1				20
TC-2 [	*T1*			1	1	1	10
	T3			1	1		30

Figure 3-2 Re-arranged transaction / attribute matrix (2)

Actually, such an ideal diagonal block structure is not easy to determine; especially in real life situations, involving a large number of transactions and attributes. Also, an ideal diagonal block structure does not exist in most of the cases. As in matrix (3) in Figure 3-3,

attributes A5 and A6 are accessed by transactions from different sub-blocks. Attributes 4 and 6 are known as inter-sub-block attributes. Although, these inter-sub-block attributes are common and prevent the formation of an ideal diagonal sub-blocks, existing clustering algorithms cannot handle them effectively [87, 88].

		Attributes						
		AC-1		*AC-3*		*AC-2*		
								Access
Transaction		*A1*	*A3*	*A2*	*A4*	*A5*	*A6*	Freq
TC-1 [	*T2*	1	1			1		20
	T4	1	1				1	40
TC-2 [	*T1*			1	1	1	1	10
	T3			1	1		1	10

Figure 3-3 Another transaction / attribute matrix (3)

In our proposed algorithm BTACO, the objective is to minimize the negative effect of those inter-sub-blocks attributes by treating them as the outliers, while the sub-blocks along the diagonal should maintain a high density of accesses (i.e. "1"). Since a sub-block may not have all "1"s in its block, some negative elements (represented by '0' in the transaction / attribute matrix) exist. Those negative elements would be regarded as the errors within a sub-block. In our definition, an outlier is an access that can not be grouped into any sub-blocks, while an error within a block is a "0" present in a block. The objective of our algorithm is to minimize the weighed number of outliers and to maximize the density of the sub-blocks. This could be done by grouping the more dominant accesses together. Other accesses will be the outliers if they are not suitable to be grouped any partitions. In other words, considering the whole data matrix as a full picture, the objective would be to reduce the number of outliers outside the sub-blocks and the number of errors within the sub-blocks. Also, the weighing of the outliers and errors

are treated equally, i.e., by considering each outlier and error with the same access frequency, they have the same numerical effect on the measurement.

In Matrix (3), the transaction with 40 as frequency count would be 4 times more important than the transaction with 10 as frequency count. As mentioned before, such a transaction / attribute matrix could not be re-arranged to form a set of sub-blocks. As a result, there is a trade-off in forming the sub-blocks with existence of some outliers and errors. The optimized result is shown in Matrix (4) in Figure 3-4. In this result, we can see that there are two outliers lying on the region of TC-1/AC-2. Also, an error exists in the sub-block (prevent the sub-blocks with all "1"). Total discrepancy is measured by the frequency count of outliers and errors, which results in 70 frequency counts. Matrix (5) in Figure 3-5 is another form of re-arranged matrix which has the same frequency counts in terms of total discrepancy. In this matrix, the positions of A5 and A6 are swapped and so there are 2 errors and 3 outliers resulted. It can be seen that different arrangements of the attributes and transaction patterns in the data matrix produce different partitioning schemes. In general, if a re-arranged data matrix has a diagonal block structure, the sub-blocks can be identified along the diagonal line.

Attributes

Transaction		AC-1		AC-2		AC-3		Access Freq
		A1	A3	A5	A6	A2	A4	
TC-1 [	T2	1	1	1				20
	T4	1	1		1			40
TC-2 [	T1			1	1	1	1	10
	T3				1	1	1	10

Figure 3-4 Matrix (4)

Attributes

Transaction		AC-1		AC-2		AC-3		Access Freq
		A1	A3	A6	A5	A2	A4	
TC-1 [	T2	1	1		1			20
	T4	1	1	1				40
TC-2 [	T1			1	1	1	1	10
	T3			1		1	1	10

Figure 3-5 Matrix (5)

Refer to Matrix (6) in Figure 3-6, the access frequencies of transaction T2 and T4 are swapped. In this matrix, there is only one partitioning scheme which is appropriate for our objective. In this example, it could be seen that the weightings of the outliers and errors affect the partitioning scheme based on the proposed objective. Since transaction pattern T2 has become more important, all attribute elements in T2 are grouped together as a component of the sub-block on the top-left corner.

		Attributes						
		AC-1		*AC-2*		*AC-3*		
								Access
Transaction		*A1*	*A3*	*A5*	*A6*	*A2*	*A4*	Freq
TC-1 [	*T2*	1	1	1				40
	T4	1	1		1			20
TC-2 [	*T1*			1	1	1	1	10
	T3				1	1	1	10

Figure 3-6 matrix with the access swapped (6)

In the above discussion, we focused on the vertical partitioning of relational database problem. In fact, the algorithm may be used in a wide variety of application as discussed before, such as the manufacturing cell formation problem[22, 72], the market basket analysis in the field of data mining[59, 61, 83] as well as cluster analysis of micro-array gene expression data[4, 24]. In the second part of this paper, the use of a two-dimensional clustering method would also be introduced for the improvement of XML database system. By the cluster analysis, the XML documents are grouped systematically so as to improve the efficiency of the query processing. The details of this will be discussed in the next part of this paper.

In the next section, the framework of Bi-tour Ant Colony Optimization (BTACO) would be introduced. BTACO, a modified version of Ant Colony Optimization, is specific for solving the two-dimensional clustering problem. There are two important components in the proposed algorithm; they are the method to re-arrange a matrix to form a matrix with a diagonal block structure and the method to identify the appropriate sub-blocks. These two important components will be discussed individually after the discussion of the framework of BTACO.

3.2 Framework of Bi-Tour Ant Colony Algorithm

To facilitate the two-dimensional data partitioning, a proposed algorithm, called Bi-tour Ant Colony Algorithm, abbreviated as BTACO, is introduced. BTACO is in fact an ant colony optimization based algorithm which uses two tours to find the desired solution. The high adaptability of Ant Colony Optimization Algorithm comes from the fact that ACO algorithm works well in solving the traveling salesman problem (TSP)[34, 36]. In fact, the ACO algorithm was first applied in solving the typical symmetric traveling salesman problem and has been found competitive with other evolutionary computing algorithms, such as Genetic algorithm[9, 11, 23]. For the one-dimensional typical partitioning problem, Lenstra and Kan Rinnooy [76] showed that it could be solved if the associated TSP could be solved. Therefore, the one-dimensional clustering problem could be formulated as a TSP. Cheng et al. [23] proposed the use of new genetic algorithm to solve the data partitioning problem by solving the associated TSP formulation. However, the algorithm does not work well in the two-dimensional clustering problem as it works by considering each dimension individually and so the results are not meaningful when treated simultaneously. The genetic algorithm-based partitioning method re-arranges the data matrix into the form such that the similar column and rows elements measured by Euclidean based distance. An example of the re-arranged matrix could be shown in Figure 3-7 which is based on the matrix from Navathe et

al [88]. The accesses on the data matrix are scattered around the data matrix, the result matrix is not suitable for determining the sub-blocks so as to reveal the relationship between the set of the row and column elements. In the computational experiment shown later, we will show that the result is not well compared to the result generated by our proposed algorithm

	Attribute																			
	14	2	12	13	9	3	7	10	11	17	18	16	15	20	19	4	6	5	1	8	
Transactions																					Access Frequencies
T3						1	1	1	1	1	1										50
T7						1	1	1	1	1											15
T12	1						1	1						1	1	1					10
T9	1	1							1						1			1			10
T6								1	1										1	1	15
T5																		1	1	1	15
T10					1						1	1							1		10
T13									1	1	1	1	1							1	10
T15										1	1	1	1	1	1						5
T4												1	1	1	1						50
T8	1	1	1	1							1	1	1	1							15
T2	1	1	1	1	1																50
T11		1	1	1	1	1											1	1	1		10
T14		1			1	1	1									1	1	1	1	1	5
T1																1	1	1	1	1	50

Figure 3-7 The re-arranged data matrix generated by GA-based algorithm

Actually, the similarity distance measure is based on the access pattern between the attributes and between the transactions, which is the variant of the bond energy approach. By this approach, it cannot guarantee the desirable patterns in which the underlying relationship between the set of columns and the set of objects can be revealed[7].

In order to remedy the problem, the re-arranged data matrix should be guaranteed to be in a diagonal block structure and so the matrix could be partitioned. Diagonal block structure would be a preferable pattern as it can be interpreted easily for revealing the relationship[7]. This could be done by constraining the permutation of

the rows or columns by certain order, such as the median center of the rows or columns. The rows or columns would be sorted by the certain order such that a diagonal block structure could be attained.

In general, the idea of BTACO for bi-clustering is as follows: From the above, we see that an individual distance measure for each dimension is not reliable in evaluating the whole data matrix, as the outliers and errors are not considered in the distance measure. As a result, the technique of median centering is introduced to make a diagonal block structure happen in the data matrix. However, there is no way to guarantee that the combination of TSP solution and median centering technique could result in producing good partitioning schemes in the data matrix. Therefore, the use of heuristic search algorithm is necessary to search the most appropriate partitioning scheme.

In fact, there have been many heuristic search method proposed such as Simulated Annealing[92, 93], Tabu Search[92] and Genetic Algorithm[9, 10], etc. Ant Colony Optimization algorithm is our choice because of its beneficial nature in forming the diagonal block structure. In ACO algorithm, the solution is constituted by a sequence of states which is determined by a stochastic decision rule carried out by an agent in the ACO system. Since the decision rule is a probabilistic rule for the agent to transit from one state to another state, the formation of diagonal block structure is made possible by simply restricting the states to be chosen in the decision rule. However, considering other heuristic techniques such as Tabu Search and Genetic Algorithm, since the results in their algorithm are not generated concurrently as in ACO algorithm, it is very difficult to form the feasible solutions

(i.e. the solution with diagonal block structure) in the population.

Besides the advantage of a stochastic decision rule, the reinforcement learning nature is also very important. There are two types of information available for searching the most appropriate solution: the heuristic information and the pheromone trail information. The former one uses the distance measure as the information to search the solution. The latter one uses the evaluation result as the information instead. As aforesaid, although the distance measure is not reliable, it would be useful in finding out the groups of similar objects to some extent. With the help of median centering, a re-arranged data matrix with a block structure would be formed for evaluation. The goodness-of-fit measurement is used as a measure for the quality of the re-arranged data matrix together with the partitioning scheme. The appropriate re-arranged data matrix would have an appropriate partitioning scheme. The evaluation result would be used as the pheromone trail information, i.e. the learning information. The better the solution get, the more is the chance that the same partial solution would be used in the next time. The subtle relationship between the feedback mechanism in reinforcement learning and the heuristic information provides a great help in giving out the most appropriate partitioning scheme in the bi-clustering.

BTACO deals with the data partitioning problem by solving two associated TSPs, they are the one derived from the rows and the other derived from the columns of the matrix. As aforementioned, the mere use of similarity distance measure is not adequate. Instead of solving the two associated TSPs individually, we find the solution in the second tour by means of the solution in the first tours as to guarantee

the re-arranged data matrix having a diagonal block structure. An evaluation method is necessary for evaluating the sub-blocks of the re-arranged data matrix. There are two approaches to be used, the dynamic programming approach and the heuristic approach. Both approaches uses weighted sum of error (WSE) as the measurement. The WSE is introduced to measure the number of errors and outliers (as mentioned in the previous section) so that the partitioning scheme with the smallest WSE would be the desirable scheme. Actually, the evaluated WSE would also be the score (i.e. fitness) of the solution found by the first and second tours. As the Ant Colony Optimization based algorithm has the capability of performing reinforcement learning, the score of the solution would be the feedback to the algorithm. By the use of reinforcement learning approach, the algorithm would search a better solution based on the feedback (i.e. score) from the previous turns of the algorithm and hence the solution would become more appropriate when the system continues.

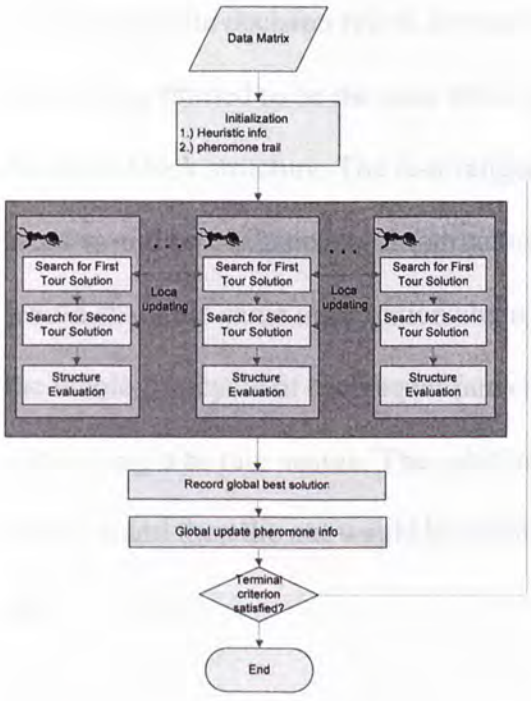


Figure 3-8 Framework of BTACO

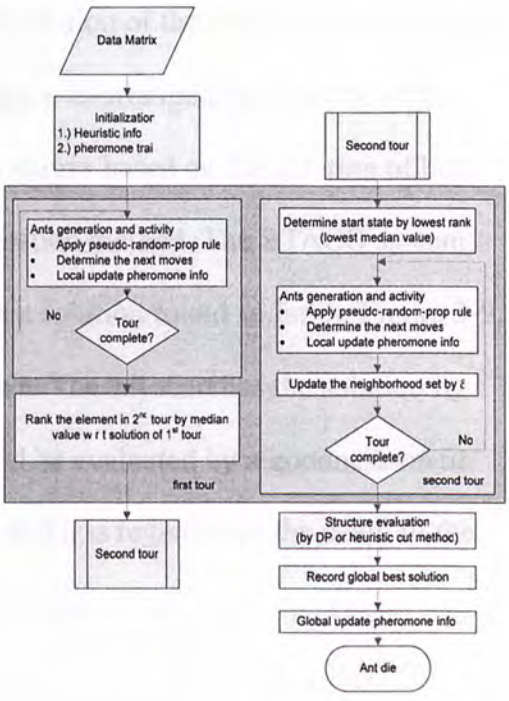


Figure 3-9 Life-cycle of each ant in BTACO

The framework of the BTACO is shown in Figure 3-8. The program flow of BTACO is given as follows: Use a data matrix as input. The algorithm has two different memory storage spaces: one for the reinforcement learning and the other for the heuristic search. The bi-tour search could be done efficiently only if the heuristic information is presumed well. In BTACO, there are a number of agents, called ants, to search the solution based on the two different information initialized. In fact, each ant has its own stochastic decision rule for searching the solution of the problem, but the rule is based on the shared heuristic and reinforcement learning information. The ants would finish the first tour as solving the TSP by ACO algorithm conventionally. The ants begin to search for the solution of the second tour only after the first tour is finished. In the second tour, the feasible neighborhood (feasible states to be chosen)

in the stochastic decision rule is limited by the solution of the first tour. The feasible choices are limited to be the ones which produce a re-arranged data matrix with a diagonal block structure. The re-arranged data matrix based on the solution of both tours would be evaluated by the structure evaluation method. The BTACO system would share the same information about the best solution found so far. In Figure 3-9, the whole life-cycle of each agent (ant) is shown. The ant searches the solution by performing a bi-tour search. The solution would be evaluated by a goodness-of-fit measure and then the ant would be eliminated and it is regarded as the death of the ant.

3.3 Re-order of the data matrix in BTACO clustering method

In this section, we would introduce the use of Ant Colony Optimization approach to re-arrange the data matrix so as to obtain a diagonal block structure. As aforementioned, the diagonal block structure of the re-arranged data matrix could facilitate the determination of the desired partitioning scheme. In a diagonal block structure, similar rows and columns are assigned along the diagonal line of the data matrix. From such a matrix, it is easy to find partitions (sub-blocks) along the diagonal line of the re-arranged data matrix at the desired position. The partitioning scheme provides a one-to-one correspondence among rows and columns. The collection of the set of one-to-one correspondence elements could be materialized to be the partitioning scheme for the data partitioning problem. In our algorithm, the

process of re-ordering of the data matrix is done by a proposed algorithm called Bi-Tour Ant Colony Optimization (BTACO) algorithm. It is a modified Ant Colony Optimization algorithm which could give out the desired row and column order simultaneously by searching different permutation orders of rows and columns. It is important to note that the identification of data partitioning scheme could not be found by the BTACO directly. The identification of sub-blocks in the data matrix is of vital in identifying of the data partitioning scheme as the re-arranged order gives us no information in how to cluster the data matrix in its best way. The detail of the identification of sub-blocks in the data matrix would be discussed in the next section.

Since the proposed algorithm is Ant Colony Optimization (ACO) based, the ACO algorithm would first be introduced as follows. The Bi-tour Ant Colony Optimization, a modified version of the ACO algorithm, would be introduced later.

3.3.1 Review of Ant Colony Optimization

1. (Initialization)
   Define number of ants k
   Initialize τ_{ij} and η_{ij} for all i,j States defined in the problem
2. (Construction)
   For each ant k do
      Repeat
         Apply probabilistic state transition rule to determine the move.
      Until ant k has completed the solution
   End for
3. (Trail update)
   For each ant k do
      Evaluate the solution by ant k $L_k(t)$
      Amount of pheromone deposited $\Delta\tau^k(t) = 1/L_k(t)$
      Update the pheromone info. by $\Delta\tau^k(t)$
   End for
4. (Terminating condition)
   If not(end test) repeat 2 - 3

Figure 3-10 Pseudo code for Ant System

1. (Initialization)
   Define number of ants k
   Initialize τ_{ij} and η_{ij} for all i,j States defined in the problem
   (Set $\tau_{ij}=\tau_0$ and η_{ij} as Ant System)
2. (Construction)
   Repeat
      For each ant k do
         Apply pseudo-random-proportional rule to determine the move.
      End-for
      (Local updating rule)
      For each ant k do
         Update $\tau_{ij} \leftarrow (1-\rho) \cdot \tau_{ij} + \rho\Delta\tau_{ij}$, where $\Delta\tau_{ij}=\tau_0$
      End-for
   Until ants have completed the solution
3. (Global updating rule)
   For each ant k do
      Evaluate the solution by ant k $L_k(t)$
   End for
   Evaluate L_{best}
   /* Update pheromone belonging to L_{best} */
   For each pair of pheromone trails involved in L_{best}
      Update the pheromone trail by $\Delta\tau^k(t)$, where $\Delta\tau^k(t) = 1/L_k(t)$
4. (Terminating condition)
   If not(end test) repeat 2 - 3

Figure 3-11 Pseudo code for Ant Colony System

Ant Colony Optimization algorithm[5, 34-36] is a bio-inspired meta-heuristic approach which has been originally used for solving one of the typical combinatorial problem, the traveling salesman problem(TSP). The approach was later developed successfully in solving many difficult combinatorial optimization problems, significant problems such as sequential ordering problem[46], vehicle routing problem[14] and quadratic assignment problem[45].

3.3.1.1 Ant System

The first version of the ant colony optimization algorithm is called Ant System (AS) [5, 34-36]. It is the prototype of a number of ant algorithms which collectively implement the ACO paradigm. The framework of the AS is given as follow: Given that there are k agents, i.e. the ants, positioning at different starting states of the problem. For each ant, it moves through the adjacent states of a particular problem by moving through neighbor nodes of the feasible states defined for solving the problem. These states constitute the partial solution of the problem. The agent moves by applying a stochastic local decision policy which makes use of transition state information. The transition state information includes the heuristic information and the pheromone information for each possible state. Once the solution is completed by the agent (ant), the agent evaluates the solution and deposit the information about its fitness as the pheromone information. The pheromone information will direct the search of the future ants to a better solution.

For illustration purpose, we use traveling salesman problem as an example to discuss the Ant System. The states are the remaining cities to which the agents can move. Each agent generates a solution by iteratively choosing a state according to a probabilistic state transition rule which is derived from the heuristic information and the intensity of the artificial pheromone on trails. It is formulated as follow:

$$P_{ij} = \begin{cases} \dfrac{[\tau_{ij}(t)]^{\alpha}[\eta_{ij}(t)]^{\beta}}{\sum_{l \in N_i}[\tau_{il}(t)]^{\alpha}[\eta_{il}(t)]^{\beta}} & \forall j \in N_i \\ 0 & otherwise \end{cases} \qquad (3.1)$$

The formula defined the probability with which an ant in state i chooses to move to

state *j*. In the above equation(3.1), τ represents the amount of pheromone deposited, η is the heuristic information which is the inverse of the distance (defined as δ(i,j)) from state *i* to state *j*. α and β are the parameters to balance the relative importance of pheromone information versus priori distance information. N_i is the set of all feasible neighborhoods when the ant is in state *i*, in which N_i could be the set of all unvisited cities. With this equation, we could get the tour in which the paths chosen having a shorter distance and are favorable based on the past information. The ants will stop searching when all the cities are visited once as the solution is formed.

Once the feasible solution is formed, a pheromone updating rule is applied which stores the information of how good of each solution is completed by the agents. The higher the intensity of the pheromone deposited (higher numerical value), the better is the path the ant passes. The amount of pheromone deposited by each ant k is defined as follows:

$$\Delta\tau^k(t) = \frac{1}{L_k(t)}, \forall k = 1,\ldots,m \qquad (3.2)$$

where $L_k(t)$ is the evaluation measure of the solution by ant k in iteration (t) and m is the number of ants used throughout the process. The updated amount of pheromone for the edge between state *i* and state *j* (l_{ij}):

$$\tau_{ij}(t) \leftarrow \tau_{ij}(t) + \Delta\tau^k(t), \forall l_{ij} \in \psi^k(t), k = 1,\ldots,m \qquad (3.3)$$

After all the ants have completed the solutions, pheromone evaporation process is applied to diversify the pheromone deposited by the ants globally. It is formulated as follows:

$$\tau_{ij}(t) \leftarrow (1-\rho)\tau_{ij}(t) \qquad (3.4)$$

where $\rho \in (0,1]$ is the pheromone trail decay coefficient. The equation is set with smoothing factor (in this case, which is ρ) so that the past pheromone information could also be considered and so as to eliminate the chance of trapping into the sub-optimal solution.

In fact, pheromone evaporation is a vital process in avoiding early convergence of the algorithm towards a sub-optimal region. The algorithm is then repeated until a termination criterion is reached. In this case, the criterion is defined as the number of the iterations in the algorithm to be reached to a desired number. The pseudo code in Figure 3-10 shows how the flow of the AS algorithm.

3.3.1.2 Ant Colony System

Ant Colony System (ACS) is the descendant of the ant system (AS) which was motivated to be built by two reasons, they are:

1. To improve the performance of the algorithm.
2. To have a better control on the balance of exploration and exploitation of solution.

Dorigo and Gambardella [34] proposed a modified Ant Colony Optimization algorithm which integrates the idea from Q-learning[65]. ACS differs from the ancestor Ant System in three main aspects; they are state transition rule, global updating rule and local updating rule.

A. ACS State Transition Rule

In ACS, a new stochastic decision rule is used to decide which next states to move to. With the new rule, the new state is chosen with a balance between exploitation and exploration. The balance between the exploitation and exploration is determined by a parameter which is between 0 and 1. For exploitation, the rule chooses the new states based on the priori and accumulated knowledge from the heuristic information and the pheromone information. Meanwhile, for exploration, the rule chooses the new state based on the probabilistic state transition rule as stated in Equation(3.1) and so it is called as biased exploration.

It is formulated as follows:

$$S_{ij} = \begin{cases} \arg\ \max_{j \in N_i} \left\{ \tau_{ij} \cdot \eta_{ij}^{\ \beta} \right\}, \text{ if } q \le q_0 \text{(exploitation)} \\ P_{ij}, \text{ otherwise (biased exploration)} \end{cases} \qquad (3.5)$$

where q is a uniformly distributed random number from 0 to 1, q_0 is a parameter ranged from 0 to 1, and S_{ij} is the probability of transiting from state *i* to state *j* given that *j* is an element of the feasible neighborhood N_i. The relative importance of exploitation vs. exploration could be determined by q_0.

Pheromone information update

The other two aspects are concerned with the pheromone information updating. In Ant Colony System, the functionalities of pheromone updating and pheromone evaporation are embedded in the global updating rule and the local updating rule.

B. ACS Global Updating Rule

Different from Ant System, only is the ant with the globally best solution[1] allowed to deposit pheromone. Together with the stochastic decision rule, the search could be more directed to the better solution and possibly the optimal solution. The global updating is performed by applying the following formula:

$$\tau_{ij} \leftarrow (1-\alpha)\cdot\tau_{ij} + \alpha\cdot\Delta\tau_{ij} \qquad (3.6)$$

where

$$\Delta\tau_{ij} = \begin{cases} (L_{gb})^{-1}, if \text{ transition i to j } \in \text{gloabl-best-tour} \\ 0, \text{ otherwise} \end{cases}$$

In ACS, α is the smoothing parameter for global updating and L_{gb} is the best evaluation measure of the solution (Assume that the objective of the problem is to minimum the L_{gb}). In the formula, smoothing factor is again being introduced instead of accumulating the pheromone directly so as to prevent early convergence. The smoothing parameter α , is usually set as a small fraction number, say 0.1 to facilitate the smoothing effect.

C. ACS Local Updating Rule

Besides updating the pheromone information globally, the pheromone level is changed by local updating rule when the ant visit the transition from state *i* to state *j*. The formulation is given as follow:

$$\tau_{ij} \leftarrow (1-\rho)\cdot\tau_{ij} + \rho\Delta\tau_{ij} \qquad (3.7)$$

where $0<\rho<1$

[1] In ACO algorithm, the globally best solution is defined as the best solution found from the start of the algorithm. It does not mean as the optimal solution of the problem.

The value of $\Delta\tau_{ij}$ is set to the initial pheromone level

$$\Delta\tau_{ij} = \tau_0, \text{ where } \tau_0 = 1/L(0) \qquad (3.8)$$

where L(0) would be the evaluation of the solution found by the execution of the ACS without the use of pheromone component. The setup is equivalent to the probabilistic nearest neighbor heuristic search. In this way, the desirability of the transition changes dynamically. Whenever a particular transition is in favor by a particular ant, the transition would become slightly less desirable next time. There is also another choice in determining the value of $\Delta\tau_{ij}$. This is inspired by Q-learning[44], that is an algorithm to solve reinforcement leaning problems[65]. The value would be set to the discounted evaluation of the next state value:

$$\Delta\tau_{ij} = \gamma \cdot \max_{z \in N_s} \tau_{ij}$$

The intuitive idea of applying Q-learning is that as the pheromone information aims at providing the information about how good a solution gets in transiting from one state to another state. The way to find the solution is similar to a reinforcement learning problem and so Gambardella and Dorigo [44] have proposed to apply Q-learning to the ant colony optimization algorithm, which is called Ant-Q as a modified version of Ant System. However, as Gambardella and Dorigo [34] mentioned, the use of initial pheromone level is in favour as it requires less computational time for each agent.

In Figure 3-11, the pseudo code of Ant Colony System is shown. It could be seen that the flow of the algorithm is slightly different from the one for Ant System. The local updating rule is applied immediately after the ants have passed through from one state to another state, which is different from AS in which the update is applied

only when all the ants have accomplished the solution (off-line pheromone update). The global update is only applicable for the best solution throughout the process of ACO algorithm. In this way, the algorithm provides a balance between the exploitation and exploration in the search of the solution for the particular problem.

3.3.2 Bi-Tour Ant Colony Optimization

In this section, the notion of Bi-Tour Ant Colony Optimization (BTACO) would be introduced to solve the data partitioning problem. As aforementioned, in our algorithm, the data matrix would be re-arranged to a desired structure for data partitioning. In fact, the structure of the data matrix is composed of the row order and the column order. The re-ordering of both rows and columns would result in different structure of the data matrix. In our algorithm, the rows and columns would be re-ordered to produce a diagonal block structure of the data matrix and so the groups of row and column elements with high similarity are formed along the diagonal line of the data matrix. In BTACO, as explained literally, there would be two tours to solve the data partitioning problem. In each iteration of BTACO, each agent (ant) has to complete both row and column tour so as to provide a feasible solution for the data partitioning problem. In fact, the feasible solution is a permutation order of rows and column, which produces a diagonal block structure. In this paper, we would use the ACS version of ACO algorithm instead of the AS version. As compared with the result in [35], the AS version of BTACO works worse than the ACS version, this also follows the result found by [34], in which ACS proves its ability in finding a better solution. The details of BTACO are shown as

follows:

Given a data partitioning problem for a m X n matrix, where n < m. In the initialization phrase, we have to initialize the heuristic information η_{ij} and the pheromone information τ_{ij} for the ants. Since there are two different tours: the row tour and the column tour, there are two sets of heuristic and pheromone information, separated by third dimension index 0 and 1.As a result, there are totally four sets of learning information, they are $\eta[0]_{ij}$, $\eta[1]_{ij}$ $\tau[0]_{ij}$ and $\tau[1]_{ij}$. For the heuristic information, the value could be the reciprocal of the similarity measure. Similarity measure here defined as the similar the pair of the objects, the lower is the value of the measure and vice versa. In this way, the heuristic values are set similarly to the ones set in typical ACO algorithm in solving the traveling salesman problem. The longer the similarity distance, the lower is the chance of being chosen in the movement of the ant when transiting from particular pair of states. There are many typical similarity measures proposed in information retrieval literatures, such as Euclidean distance measure, Manhattan distance measure, Jaccard coefficient and cosine function[83], etc. As in the case of vertical partitioning problem, the similarity measure uses the Manhattan distance measure[23, 76]. The distance measures between a pair of rows (transactions) and columns (attributes) are formulated as follows:

For transaction,

$$d_{ij}^{Transaction} = \sum_{k=1}^{n} \begin{cases} \left| a_{ik} \times F_i - a_{jk} \times F_j \right| & if\ a_{ik} \neq a_{jk} \\ 0 & ,otherwise \end{cases} \qquad (3.9)$$

For attribute,

$$d_{ij}^{Atribute} = \sum_{k=1}^{m} \left| (a_{ki} - a_{kj}) \times F_k \right| \qquad (3.10)$$

where a_{ij} represents the entry in the unorganized transaction-attribute matrix. i represents transaction T_i and k represents attribute A_k. F_i is the access frequencies for transaction T_i per unit time period. By setting index 0 represents the row order and index 1 represents the column order, the heuristic information is initialized:

For transaction,

$$\eta[0]_{ij} = 1/d_{ij}^{Transition} \qquad (3.11)$$

For attribute,

$$\eta[1]_{ij} = 1/d_{ij}^{Attribute} \qquad (3.12)$$

For the pheromone trail information, we define the notation in the same sense. The pheromone information for the row order and column order are stored in $\tau[0]_{ij}$ and $\tau[1]_{ij}$ respectively. It is important to note that the pheromone information for all pair of transition should be set as the very small value instead of zero initially in order to let the state transition rule to function properly (recalling back to equation (3.5) and (3.1)). Otherwise, the stochastic decision rule will be undefined.

After the initialization, we have to set the starting state for triggering the algorithm to use the state transition rule to complete the solution. In the original Ant Colony System (ACS) and Ant System (AS)[34-36], they are proposed to apply in solving the traveling salesman problem. The TSP has the property that in each solution, the choice of starting state should not alter the quality of the solution as the objective of

TSP is to form a cyclic graph which connects all the nodes in the given graph. No matter which starting node is, the optimal solution should still be resulted. As a result, in ACS or AS, they can choose to set the starting state either in randomly way or at all the available nodes. Setting the starting state randomly is not desirable in the data partitioning case, as each starting state would result in a different structure in the data matrix. Also, since there are two tours to be completed in BTACO, for the second tour, setting the starting state arbitrary is not appropriate. The starting state should result the re-arranged data matrix in the diagonal block structure. The way to choose such a row (or column) will be discussed later. Because of the above issues, for the first tour, we have to set each agent to start at each possible state, and so the number of agents should be the multiple of the possible states in either row or column order. For the second tour, we would choose the elements which have the most of the elements assigned in the top-left corner of the re-arranged data matrix. Also, as there is no limitation in choosing either row or column order as the first tour. We have to start the algorithm by setting the agents (ants) as the width and height of the data matrix, with each ant starts the algorithm at a different row and column position. Since some of the ants begin at the row tour while the others begin at the column tour, there is a subtle relationship of learning between the ants. In ACS, the local updating rule is triggered whenever the ant moves from one state to another state. The pheromone trail information could be used by the ants whenever they are finding out the solution in the second tour. On the other hand, the AS would only update the pheromone trail only when the solutions are completed by the ants. The asynchronous and concurrent learning effect within each iteration makes the ACS superior than the AS in our problem.

For the first tour, no matter which orders the ants are in, they could complete the solution as in the typical TSP case. Given a starting state, each ant chooses the state to be moved using by the stochastic decision rule as discussed. The pheromone information of the movement is also updated immediately by the local updating rule. The set of neighborhood would also be updated similarly as in the TSP, i.e. the set of unvisited states in the problem definition. The solutions in the first tour would be finished when all the states, i.e. all the possible rows (or columns) are chosen once.

As aforementioned, for the second tour, we have to find the starting state which fulfills the requirement of building diagonal block structure. In fact, through the search of the solution in the second tour, it is necessary to find a solution such that the complete solution (i.e. the solution combining both the solution from first and second tours) should be in diagonal block structure which can facilitate the block identification as being described in the next section. In order to do so, we perform a median / mean centering of all the candidates and rank them in the ascending order of the median value (discussed later). By choosing the row (or column) with the smallest rank (the one with smallest median from left to right, or from top to bottom), the structure of the re-arranged matrix could be in desired form.

	5	4	2	1	6	3
2	X	X				
4	X	X	X			
1	X		X		X	
3			X			X

Figure 3-12 Median Centering Technique

Actually, limiting the choices of starting states is not enough. We have to further

control the neighborhood sets of each just-visited state for the ants to search. For the vertical partitioning case, Cheng et al[23] has proposed using Genetic algorithm-based partitioning method which clusters the transactions and attributes individually. The result of the re-arranged matrix is not in a diagonal block structure (as shown in Figure 3-7 in the previous section) and so it is difficult to identify the relationship among the transactions and the attributes. In fact, the undesired structure is due to the outliers in the data matrix. As the distance measure used in the genetic-algorithm is based on the elements in the data matrix, the outliers are also counted in the distance measure. These outliers bias the result of the re-arranged matrix and cause the structure of the re-arranged data difficult to show out the relationship. As a result, a reference is necessary for re-arranging the data matrix meaningfully. In fact, for the partitioning problem involving two dimensional data, For the block clustering method proposed in [37, 54], they deal with the problem by sorting the second dimension data in certain order, such as the year and time frame, etc. However, as some of the data does not have the centering reference, such as the clustering of micro-array gene expression data and the vertical partitioning. King [69] have proposed the "Rank-Order Clustering" (ROC) approach for forming diagonal block structure in a heuristic way. In his approach, the column (row) entries are treated as binary number and the solution is the sort order of the entries in descending order. If the matrix have nice diagonal block structure, the approach could result the structure with consecutive "1"s are assigned inside the sub-blocks. However, this is not ideal in most of the cases as discussed before. In our approach, we sort the entries by their center elements based on the sequence of the solution in the first tour instead. This avoids the sorting being based on the pattern of "1" allocated. The sorting method works as follows: Given that the first tour is

completed, all the candidates for the second tour are centered by the order of the solution in the first tour. The median centering value of each element in the second tour is the average position of positive correspondence (marked as "X" in Figure 3-12) w.r.t. to the order solution by the first tour. The mathematical notation of defining median center of row *i* is:

$$median_{row\ i} = \underset{m \le col}{\arg}\{\sum_{j \le m} a_{ij} = \left\lceil \frac{1}{2} \sum_{j \le col} a_{ij} \right\rceil\} \qquad (3.13)$$

where *col* is the length of the column order, and we assume that $a_{ij} = 1$ if the positive correspondence exists or = 0 otherwise. The notation for median of column order could also be set similarly. Refer to Figure 3-12, there is a re-arranged matrix with the permutation of column order as the first tour. The row order has already been sorted by the median centering reference value defined by equation(3.13). For instance, considering the third row (with index *1*), it has 3 positive correspondence elements at 1st, 3rd and 5th positions w.r.t the column order. By the equation(3.13), we get the median value of 3. At the same time, considering the fourth row (with index 3), the median value is of 3 too. However, the row order should be placed in such way as by comparing with both rows with index *1* and *3*, the former one have a higher similarity with the second row (with index *4*). The objective of grouping similar rows together is also considered in this arrangement. As a result, for the second tour, the neighborhood set in the stochastic decision rule is constrained to be the set of the unvisited states such that the median centering value is bounded by the neighborhood bound of the current state. The neighborhood set would be dynamically updated whenever the neighborhood bound is changed. In fact, the use

of neighborhood bound is to control the restrictiveness of the tour to be in median centering order. The user parameter δ (ranged from 0 to 1) is used to adjust the restrictiveness. The smaller the value of δ, the higher is the restrictiveness. For defining the value of δ, we first introduce the row (or column) score. The score is actually the rank of the rows based on the median centering order. The neighborhood could then be defined as:

$$\textit{Neighborhood set}=\{N \mid \forall N \in s, \text{ s.t. s.median} \leq \text{current.median}+\delta \cdot \text{max.score} \cap s \notin \textit{partial\_tour}\} \quad (3.14)$$

where

s.median is the median value of the state s

current.median is the median value of the current state

max.score is the total number of median value identified in the tour

The equation means that the neighborhood set should be the collection of elements such as the median value of them should be bounded by the maximum threshold of the median value of current state. Also, the set should include the unvisited states only. In this way, the ant could choose the one with high similarity w.r.t the current state while the solution is roughly ordered by the median centering method. δ is actually the degree of controlling the k-nearest upper-bound neighbor to be included in the neighborhood set. If δ is set to 0, it is often that the neighborhood set is empty during the iteration of second tour. Hence, it is necessary to set one rule to test whether the set is empty. If it is so, the neighborhood set is updated by the elements with one more rank of the current state. In fact, there is a trade-off between the similarity measure and the median centering value in determining the movement to the next states. The most restrictive neighborhood set is not necessary to be the best strategy as the choice of similar element is limited. The computational study of the effect of d would be shown in the experiment part. In Figure 3-7, the framework of

BTACO is shown. In BTACO, the evaluation of the block structure for global updating rule would be discussed in the next chapter as it involves other issues in order to evaluate the structure efficiently. The amount of pheromone to be updated globally would be the reciprocal of the value of the evaluation resulted. For the local updating rule, the amount of pheromone would also be the value of the evaluation resulted by the nearest heuristic search method It is a method which uses the heuristic information only to permute the row and column order.

3.4 Determination of partitioning scheme

In the last section, we have discussed about how to permute the order of rows and columns to form the diagonal block structure by BTACO. In fact, the formation of a diagonal block structure aims at supporting the identification of sub-block structure only, which rows and columns in the matrix should be grouped together have not been determined. Since a diagonal block structure does provide a clue in partitioning the data, a partitioning method is proposed to identify the sub-blocks in the re-arranged matrix. In our algorithm, the proposed partitioning method provides two functions in our proposed algorithm:

- To identify the sub-blocks in the matrix, i.e. which rows and columns should be grouped together.
- To evaluate the fitness of the re-arranged matrix for grouping rows and columns so as to update the pheromone information in the ant-colony

optimization based part of the algorithm.

As the fitness of identifying sub-blocks is equivalent to the fitness of partitioning the data, the partitioning method directly tell us the result of partitioning the data by the identification of sub-blocks. For a given permutation order of rows and columns, the partitioning method give out the information about which rows and columns should be grouped together by means of a partitioning scheme. Since the partitioning method finds the sub-blocks based on certain re-arranged order of the matrix, the partitioning scheme varies on the fitness of the re-arranged data matrix. A fit re-arranged data matrix becomes crucial in getting a good data partitioning result. In fact, recalling back to the previous section, the fitness of the permutation order depends on the heuristic information and the pheromone information, in which the latter information is in turn updated by the goodness of the data partition resulted. On the other words, the permutation order explored depends on how well of the data partitioning scheme given by the partitioning method.

From the above, we can see that the objective of the partitioning method is to find out the data partitioning scheme and to use the scheme for evaluating of the permutation order of the data matrix. In general, there are three issues arisen about getting a good data partitioning scheme:

- The permutation order of the data matrix
- The evaluation function of the data partitioning scheme
- The exploration method of the most suitable scheme

First, as we have discussed before, since the re-arranged matrix provides a clue for data partitioning, a fit permutation order is necessary for getting a good data

partitioning scheme. As our algorithm is Ant Colony Optimization based, it learns from the result of the partitioning scheme and gives out better permutation order as the algorithm iterates. Second, a measure for the fitness of the data partitioning scheme is required for the evaluation of the scheme resulted by the partitioning method. It is important to note that the result of the evaluation function should be consistent with different permutation order of the matrix. In the other words, Comparing with two re-arranged data matrix with worse and better permutation order respectively, based on the fittest partitioning scheme, the former one should have poor rating than the latter one. Otherwise, the rating cannot be used to determine the update of the amount of pheromone information in the ACO part and we cannot determine which permutation order is more suitable for the partitioning method. Finally, since there are numerous possible partitioning schemes, say, suppose we have a n X m matrix, where n < m, then there would be $\sum_{k=0}^{n-2}(k+1)!\times(C_k^{n-1}\times C_k^{m-1})$ number of feasible partitioning scheme. The formula is derived as follow:

Suppose: We have a n X m matrix, where n < m.
There are (n-1) possible cuts along the row and (m-1) possible cuts along the column.
For partitioning the data matrix into c^2 parts:

c=1 No. of scheme = $1!\times C_0^{n-1}\times C_0^{m-1}=1$

(The result is trivial as the whole data matrix is treated as the only one possible partitioning result)

c=2 No. of scheme = $2!\times C_1^{n-1}\times C_1^{m-1}=2\times(n-1)\times(m-1)$

(For partitioning the data matrix into 4 parts, we have to consider all the combination of each row and column division once. Factorial of 2 indicates that we have to consider all the combination of each one-to-one correspondence of the sub-matrix)

c=3 No. of scheme = $3!\times C_2^{n-1}\times C_2^{m-1}=3!\times(n-1)(n-2)/2!\times(m-1)(m-2)/2!$

....

c=n-1 No. of scheme = $n!\times C_n^n\times C_n^m=1\times(m-1)(m-2)....(m-n+1)$

(Since there are n rows only and n < m, there can be up to $(n-1)^2$ parts divided in the data matrix. Also, as m > n, there are more than one possible scheme as above.)

As a result, the total number of possible scheme = $\sum_{k=0}^{n-2}(k+1)!\times C_k^{n-1}\times C_k^{m-1}$

For instance, if we have a 15 x 20 matrix, there would be 7x 10^{16} (79,297,526,011,133,968). Even we have re-arranged the matrix to be in a diagonal block structure (and hence the cost of permutation is elimination by assigning the one-to-correspondence along the diagonal line only), there would still have 800 millions (818,797,572) feasible partitioning schemes. The exhaustive enumeration is not possible even for today's CPU power. Even the dynamic programming approach is not efficient enough for such an enormous number of possible partitioning schemes. As discussed later, an efficient and scalable partitioning method is proposed in our algorithm. In the experiment, we would also show that the proposed method is competitive with the dynamic programming approach.

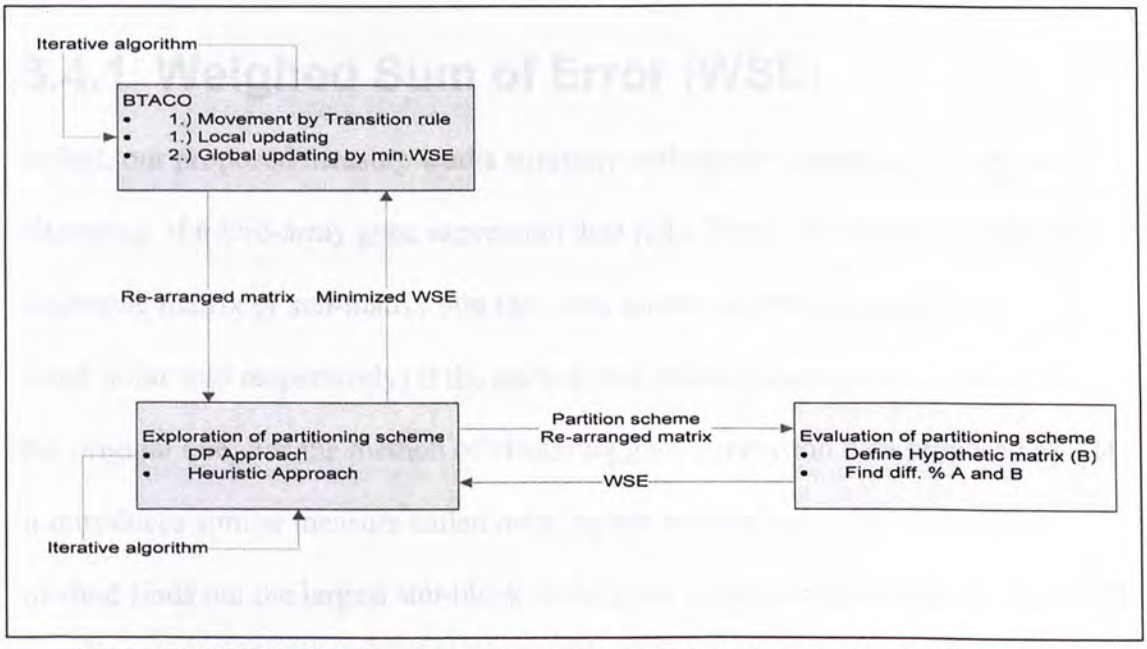


Figure 3-13 Relationship between ant algorithm, evaluation scheme and exploration scheme

In general, the relationship between the three issues is shown as in Figure 3-13. As seen from the flow diagram, the permutation order is outputted from the ACO

algorithm and has been discussed. In the following, we would first introduce the proposed evaluation function for data partitioning and then we would show the ways of applying the evaluation function to explore the possible partitioning scheme.

There have been many evaluation measures proposed for data partitioning as discussed in the previous section. However, the measures such as the Euclidean distance, Jaccard coefficient and cosine function[83] are suitable for partitioning the data in one dimension only. As we have to partition the data in two-dimension, they are not suitable at all. The experiment part would show the incapability of these measures to locate which row and column elements would be grouped together. Because of this, we introduce weighed sum of error (WSE) as a measure for data partitioning.

3.4.1 Weighed Sum of Error (WSE)

In fact, our proposed measure works similarly with direct clustering [37, 54] and clustering of micro-array gene expression data [24]. Direct clustering method splits the whole matrix or sub-matrix into two (also known as free-point split and fixed-point split respectively) if the revised root-mean-square error is smaller than the original one. For the method of clustering gene expression data proposed by [24], it introduces similar measure called mean square residue score (H), in which the method finds out the largest sub-block in the gene expression data with the score (H) satisfying a certain threshold δ, through a sequence of insertion and removal of row and column. For the both clustering methods above, the results found are not necessary to be mutually exclusive cluster set. On the other hand, the method

proposed by [24] can only find one sub-block of gene expression data at each time, in which the conditions and genes may have overlapping relationship. Recalling back to our algorithm, we would like to identify the non-overlapping partitioning of rows and columns, but the above methods can only measure the homogeneity of the formed clusters in the data matrix. Rather, we also have to measure the degree of homogeneity between clusters, so as to measure the fitness of the clustering scheme. There have been many indices proposed to find out the measure satisfying aforementioned criteria, but the indices are introduced for the traditional one-dimension clustering method, such as hierarchical clustering. Therefore, we introduce the Weighed Sum of Error (WSE) which can measure the homogeneity within and between clusters and is suitable for a two-dimension problem.

For our proposed measurement, we derive the idea from the goodness-of-fit test which is a typical statistical hypothesis test. For the goodness-of-fit test, we test whether the sampled data fits certain theoretical distribution function. Analogous to the goodness-of-fit test, we assign a hypothetic matrix to the re-arranged data matrix and evaluate the degree of the goodness-of-fit. We evaluate the goodness-of-fit by calculating the Weighed Sum of Error as the sum of the difference of each value in the element of the hypothetic matrix and the re-arranged data matrix (Observed data matrix). For instance, WSE can be formulated as:

$$WSE = \sum_{i=1}^{m}\sum_{j=1}^{n}\left|a_{ij} - A_{ij}\right| \qquad (3.15)$$

where a_{ij} is the element in the *i*th row and *j*th column of the re-arranged data matrix and A_{ij} is the element in the *i*th row and *j*th column of the hypothetic matrix. The smaller the value of WSE, the more is the number of matches between the element

of both matrices and hence the higher the degree of the goodness-of-fit. In fact, sum of square error is also common in calculating the difference between the hypothetic value and the observed value in the statistical study.

In several scenarios, each element in the data matrix is not treated equally. As illustrated, for the transaction / attribute matrix in the vertical partitioning problem, each row (transaction) is weighed by the access frequency of the transaction pattern. As a result, with the same number of difference of elements between two rows, the one with the higher access frequency should be assigned with a larger sum of error than the one with the lower access frequency. The formula (3.15) is modified as:

$$WSE = \sum_{i=1}^{m}\sum_{j=1}^{n}(\left|a_{ij} - A_{ij}\right| \times \frac{F_i}{F_{\max}}) \qquad (3.16)$$

where F_i is the access frequency of row i and F_{max} is the maximum access frequency among all the rows in the data matrix. F_{max} normalizes the weight to be within 0 and 1 for calculating normalized WSE. In general, weighed sum of error can be formulated as:

$$WSE = \sum_{i=1}^{m}\sum_{j=1}^{n}(\left|a_{ij} - A_{ij}\right| \times \frac{W_{ij}}{W_{\max}}) \qquad (3.17)$$

where W_{ij} is the weight for the element in ith row and jth column of the data matrix. W_{max} is the maximum weight found in the matrix in which it normalizes the weight as the same way in the previous equation.

3.4.2 Materialization of partitioning scheme via hypothetic matrix

Although the function for evaluating the hypothetic matrix is defined, it is also

necessary to materialize the partitioning scheme via the generation of appropriate hypothetic matrix. The partitioning scheme could be materialized by forming a hypothetic matrix in which element with '1' in the data matrix indicates the correspondence of the row and column and element with '0' indicates the negative correspondence. In our proposed algorithm, since we are dealing with the non-overlapping partitioning problem, the rows and columns should have one-to-one correspondence to each other. Besides, as the re-arranged data matrix is in a diagonal form structure, the one-to-one correspondences are ordered from the left-top corner to the right-bottom corner, the hypothetic matrix should have the sub-blocks assigned along the diagonal line from the left-top corner to the right-bottom corner. The above criteria can be satisfied by forming a hypothetic matrix with many sub-blocks (as indicated by '1' elements) along the diagonal line, which are not connected (so as to guarantee the one-to-one correspondence).

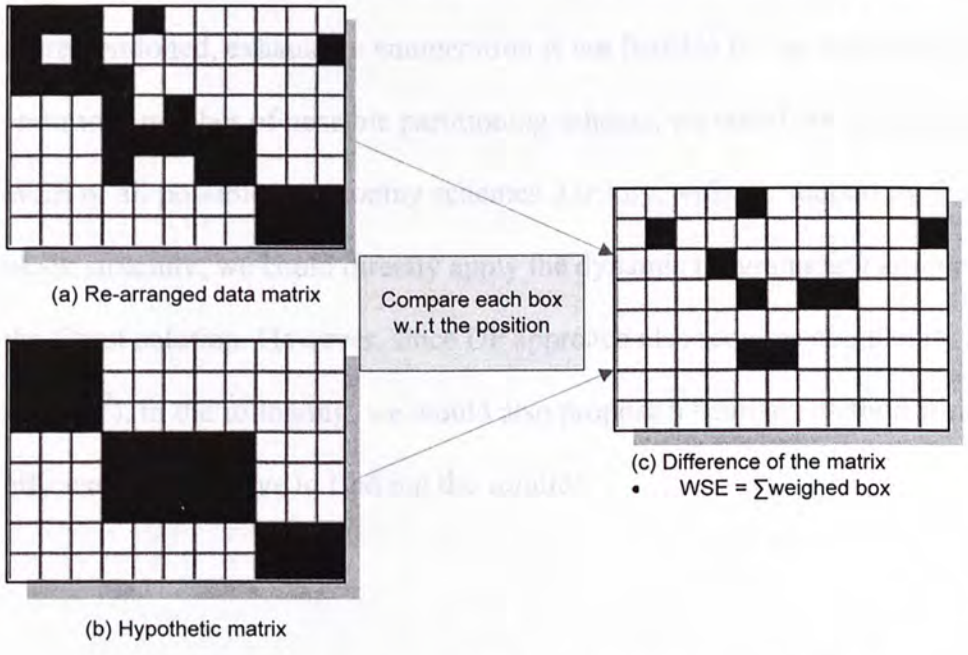


Figure 3-14 How WSE used to evaluate the partitioning scheme

In general, we have illustrated how the weighed sum or error (WSE) to be calculated in the graphical way in Figure 3-14. In Figure 3-14(c), the remained element in the matrix would be the error when comparing with the re-arranged matrix and the hypothetic matrix. The objective of the partitioning method is to minimize the weighed number of the remained element in the matrix.

3.4.3 Search of best-fit hypothetic matrix

In the last section, we have discussed about the necessary criteria for materializing the partitioning scheme through the formation of hypothetic matrix in our algorithm. In this section, we would introduce the methods for finding out the feasible hypothetic matrix and also the best-fit one with respect to the particular re-arranged data matrix. The best-fit partitioning scheme would be the one with the smallest WSE with respect to the particular re-arranged data matrix. However, as aforementioned, exhaustive enumeration is not feasible for the search as it involves enormous number of possible partitioning scheme, we could not merely calculate the WSE of all possible partitioning schemes. Luckily, with the support by the diagonal block structure, we could directly apply the dynamic programming approach to get the fittest solution. However, since DP approach also requires complexity of $O((mn)^2)$, in the following, we would also propose a heuristic method which is efficient and adaptive to find out the solution.

3.4.4 Dynamic programming approach

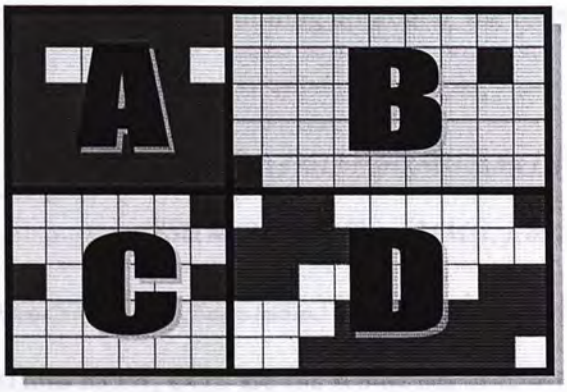


Figure 3-15 Definition of sub-block notation for easier illustration

Dynamic programming is a way of decomposing certain hard problem to smaller problems into equivalent formats that are more amenable to final solution. It is so the most direct way to deal with the exhaustive enumeration aforementioned. In our problem, we formulate the search of partitioning scheme as follow: For each matrix M (or sub-matrix), we divide it into four parts, say A, B, C and D as shown in Figure 3-15. As the best-fit partitioning scheme would be the one with the smallest weighed sum of error, we find the minimum WSE by dynamic programming approach as:

$$f^*(M) = \min_{\alpha \in A, \beta \in B, \gamma \in C} \{WSE(\alpha,1) + WSE(\beta,0) + WSE(\gamma,0) + f^*(\kappa)\} \qquad (3.18)$$

where A,B,C are all the possible set of sub-blocks, and the merge of the sub-blocks α, β, γ and κ should result in the whole block M. WSE(x,1) is defined as the weighted sum of error in which the positive correspondence is assumed in the sub-matrix x. In the same sense, WSE(x,0) is defined as the weighted sum of error in which the negative correspondence is assumed in the sub-matrix x. By this approach, we could get a set of sub-blocks which results in the smallest WSE. In fact, the set of sub-blocks indicates the one-to-one correspondence of the rows and columns. The

partitioning scheme can be reflected by the set of sub-blocks resulted. In the actual implementation of the dynamic programming approach, the formula is modified to be:

$$f^*(M(0,0,m,n)) = \min_{i \in ROW, j \in COL} \{WSE(M(0,0,i,j),1) + WSE(M(i,0,m,j),0) + WSE(M(0,j,i,n),0) + f^*(M(i,j,m,n))\} \quad (3.19)$$

where $M(y_{\text{left-top}}, x_{\text{left-top}}, y_{\text{right-bottom}}, x_{\text{right-bottom}})$ indicates the sub-block of the data matrix to be involved and ROW, COL are the whole set of row and column elements respectively. The pseudo code for the implementation of the above formula is shown in Figure 3-16.

```
Initialize:
1.  double[ ][ ] WSE = new double[0..row_of_matrix-1][0..col_of_matrix-1];
2.  partitionScheme[ ][ ] PS = new partitionScheme[0..row_of_matrix-1][0..col_of_matrix-1];
3.  for (int i=row_of_matrix; i>= 0; i--) {
4.      WSE[i][col_of_matrix-1] = WSE(i,row_of_matrix, col_of_matrix-1, col_of_matrix, 1)
5.  }
6.  for (int i=col_of_matrix; i>= 0; i--) {
7.      WSE[row_of_matrix-1][i] = WSE(row_of_matrix-1,row_of_matrix, i, col_of_matrix, 1)
8.  }

Main:
9.  for (int i=row_of_matrix-1; i>=0; i--) {
10.     for (int j=col_of_matrix-1; j>=0; j--) {
11.         eval_sub-block(i,j)
12.     }
13. }

eval_sub-block(row_start, col_start):
14. WSE_min = WSE((row_start, col_start, row_of_matrix, col_of_matrix),1)
15. for (int i=row_of_matrix-1;i>row_start;i--) {
16.     for (int j=col_of_matrix-1;j>col_start;j--) {
17.         WSE_local = WSE((row_start, col_start, i, j),1) +
18.                     WSE(row_start, j, i, col_of_matrix),0) +
19.                     WSE(i, col_start, row_of_matrix, j),0) + WSE[i][j]
20.         if (WSE_min > WSE_local) {WSE_min = WSE_local
21.             PS_min = PS[i][j] + new_partition_point(i, j)
22.     }
23. }
24. WSE[row_start][col_start] = WSE_min
25. PS[row_start][col_start] = PS_min
```

Figure 3-16 Pseudo code for dynamic programming approach

Considering the pseudo code in Figure 3-16, the algorithm consists of three parts:

Initialization, Main and evaluation of sub-block. For the first part, we initialize two two-dimension arrays with the same size of data matrix to store the optimal WSE and its partitioning scheme respectively. We also have to initialize the WSE of the atomic matrix, i.e. 1 x n or m x 1matrix so as to define the final state of DP for triggering the iterative evaluation. We trigger the DP approach by considering the atomic matrix first and considering the larger sub-matrix gradually. In the evaluation part, we first calculate the WSE of the whole sub-block as the minimum reference. We then consider each partitioning scheme ordered by the largest sub-block A to the smallest one (refer back to Figure 3-15). The partitioning scheme attaining the smallest WSE would be the best scheme for the sub-block considered.

In the direct way, we calculate the WSE by summation of the difference of hypothetic value and the observed value as in equation(3.16). In this way, the computational complexity is of O(mn), where m and n comprises the dimensions of the matrix involved. Since the complexity is not in favour for the large problem, a trade-off is made between the memory space and the computational complexity. We would give out the detail as below:

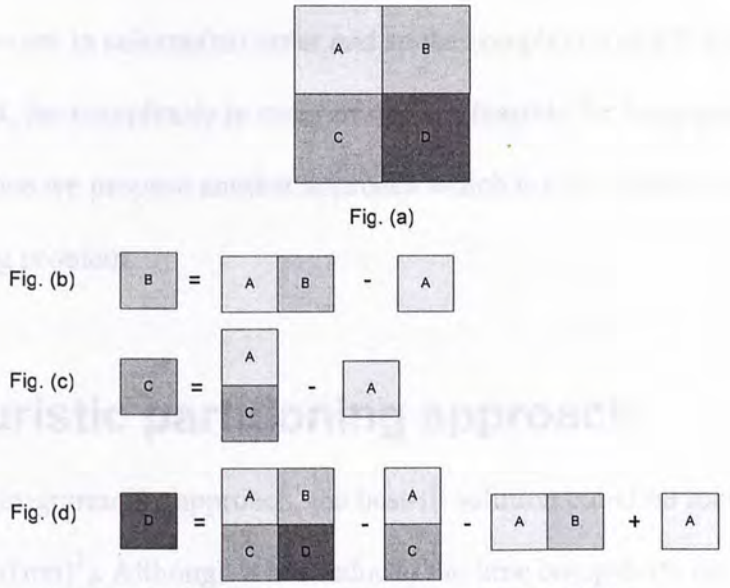


Figure 3-17 Illustration for efficient calculation of WSE

In order to facilitate the effective calculation of WSE, we have to build up two matrices, say matrix_0[0..row][0..col] and matrix_1[0..row][0..col]. matrix_1[i][j] stores the WSE of the sub-matrix with left-top=(0,0) and right-bottom=(i,j) w.r.t. positive correspondence. matrix_0[i][j] stores the WSE in the same way, except that the WSE is now w.r.t the negative correspondence. With the matrices, we could find out the WSE of any size of the sub-matrix in the constant time. In the Figure 3-17, it illustrates how we could calculate the WSE of region B, C and D by simple subtraction of the sub-blocks in diagram (b) to (d) respectively. The calculation of the WSE of region A is straightforward; it simply just refers from the matrix. We can see that even the region of D requires three binary operations only and so the calculation of WSE is very fast. Although the main drawback is that the space complexity is of O(mn), with the current configuration of computer, it is capable of storing the matrix.

In general, from the pseudo code, we can see that the core part of the algorithm involves four nested for-loops in total, in which two for-loops are in row(n) order

and the other two are in column(m) order and so the complexity of DP approach is $O((mn)^2)$. In fact, the complexity in order of 4 is not feasible for large partitioning problem and hence we propose another approach which is efficient and scalable for large partitioning problem.

3.4.5 Heuristic partitioning approach

In the dynamic programming approach, the best-fit solution could be found in the complexity of $O((mn)^2)$. Although it has reduced the time complexity enormously compared with mere exhaustive enumeration. The quadratic complexity is not efficient when the data matrix to be evaluated is large. Hence, we propose a heuristic partitioning approach in which for a n x m matrix, where $n < m$, the worst complexity is of $O(n(m+n))$, and in the average case, as it is not common to partition the whole data matrix into single row (or column) sub-matrix. The complexity is of $O(c(m+n)) \fallingdotseq O(m+n)$, when the matrix is large and the matrix can be partitioned into a relative small number of cluster, i.e. $c<<n$.

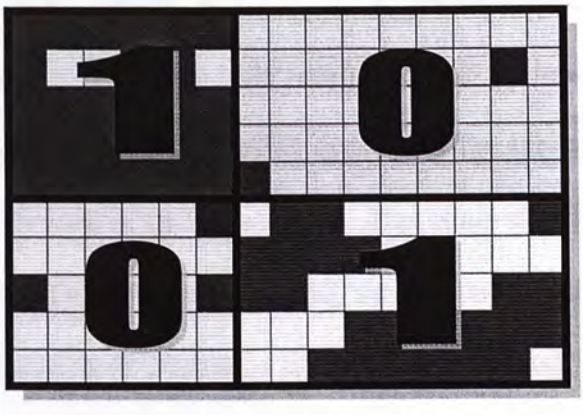


Figure 3-18 Criterion for heuristic binary partitioning approach

The heuristic partitioning approach is by means of iterative bi-section of the data matrix until there is no need to further bi-section. The criteria of bi-section

partitioning is based on the intuitive idea that whenever the re-arranged matrix is in diagonal block structure, the matrix (or sub-matrix) is bisected to form four sub-matrices as in Figure 3-15, in which the sub-matrices B and C should result with the largest sub-blocks in which the WSE with negative correspondence (i.e. WSE(:, 0)) is the smallest. The criterion is shown graphically in Figure 3-18. By the principle of divide-and-conquer, the data matrix is divided into four sub-blocks each time and only the left-top as well as the right-bottom sub-matrices would be considered in the future iteration. As a result, the algorithm is very effective in dividing the problem into smaller problems and finds optimal (near-optimal) result in the whole data matrix effectively. The details of the approach are shown as follows:

In a certain matrix M, as we have to find out the largest sub-block in left-bottom region (C) and right-top region (B), in which they have the smallest WSE w.r.t. the negative correspondence, we could not bisect the matrix M by merely testing for all possible sets of rows and columns in the particular permutation order as in dynamic programming approach. This would only result in a local optimum of bisecting the matrix M into four matrices and the further division of the sub-blocks would result in local optimum of bisecting the sub-matrices. This is not desirable as it may not result in the best partitioning scheme. Instead, we find out the partitioning scheme satisfying the aforementioned criteria by first bisecting the matrix M into two equally or (n-1)/2 and (n+1)/2 order (if the number of row or column is in odd number) sub-matrices in one particular dimension. Then we evaluate the remaining dimension by bisecting the sub-matrices further into four sub-matrices. The row or

column, say i, which results in the smallest sum of WSE[2] would be either the vertical or horizontally part of the individual partitioning scheme. Afterwards, we consider the whole matrix M again and bisect it at i which should be done in the same dimension as the remaining dimension. Again, we evaluate the first dimension by bisecting the sub-matrices further into four sub-matrices as aforementioned. The bisection position, say j, which results in the smallest sum of WSE, would be the other component of the individual partitioning scheme. Although this is the desirable individual partitioning scheme found under the matrix M, we have to test it whether it is necessary to partition the matrix (or sub-matrix) or otherwise halt the further division of the matrix. This could be done by comparing the sum of WSE of the partitioned matrix with WSE of the matrix which treat the whole block with positive correspondence. If the WSE of the partitioned matrix is smaller than the latter one, the matrix M should be further divided or otherwise the bisection should be halted. In general, for a n X m matrix, the mathematical notation of finding out the vertical and horizontal bi-section point would be:

$$\textit{vert.}\ \text{cut}=\arg\min_{c\in COL}\{\sum_{i=1}^{\alpha}\sum_{j=1}^{c}|a_{ij}-1|+\sum_{i=1}^{\alpha}\sum_{j=c}^{m}|a_{ij}-0|+\sum_{i=\alpha}^{n}\sum_{j=1}^{c}|a_{ij}-0|+\sum_{i=\alpha}^{n}\sum_{j=c}^{m}|a_{ij}-1|\} \quad (3.20)$$

where we assume that the un-weighed case as formula (3.15) is considered. COL is the possible column positions could be chosen. α is the pre-defined horizontal cut, which could be the dimension of row divided by two if the horizontal cut is not found. Otherwise, α is the horizontal cut found previously. Similarly, the horizontal bi-section point would be formulated as:

$$\textit{hori.}\ \text{cut}=\arg\min_{c\in ROW}\{\sum_{i=1}^{c}\sum_{j=1}^{\beta}|a_{ij}-1|+\sum_{i=1}^{c}\sum_{j=\beta}^{m}|a_{ij}-0|+\sum_{i=c}^{n}\sum_{j=1}^{\beta}|a_{ij}-0|+\sum_{i=c}^{n}\sum_{j=\beta}^{m}|a_{ij}-1|\} \quad (3.21)$$

[2] Sum of WSE refers to the summation of weighed sum of error of the four regions as defined in Figure 3-15

where ROW is the possible set of row positions and β is the pre-defined vertical cut which is defined similarly as the vertical bi-section point formulation.

```
Initial input: original data matrix
Output:split-matrix
      Weighted sum of error (WSE) of split-matrix
    1.  bicut (matrix) {
    2.     Evaluate WSE(matrix);
    3.     c = no. of columns in matrix / 2;
    4.     For each row (r) in matrix {
    5.        Cut the matrix into 4 regions by column c and row r;
    6.        Evaluate WSE(top-left_region, "1"); // k = 1
    7.        Evaluate WSE(top-right_region, "0"); // k = 0
    8.        Evaluate WSE(bottom-left_region, '0'); // k = 0
    9.        Evaluate WSE(bottom-right_region, '1'); // k = 1
    10.       Total WSE = sum of above four evaluation
    11.       If total WSE is mimimum, set r as the horizontal cut h.
    12.    }
    13.    For each column (c) in matrix {
    14.       Cut the matrix into 4 regions by column c and row h;
    15.       Evaluate four regions again (same as above)
    16.       Total WSE = sum of above four evaluation
    17.       If total WSE is minimum, set c as the vertical cut v.
    18.    }
    19.    If the total WSE by cross cut v & h < WSE(matrix) {
    20.       Bicut(top-left_region);
    21.       Bicut(bottom-right_region);
    22.    }
    23. }
```

Figure 3-19 Pseudo code for heuristic partitioning approach

Using the above procedure, we can efficiently find out the largest sub-blocks with negative correspondence in the left-bottom and right-top corners. By iteratively considering the left-top and right-bottom sub-matrices of the matrix M (refer to Figure 3-15 they are sub-matrices A and D respectively), we could find the desirable partitioning scheme by merging each individual partitioning scheme in each sub-matrix. The pseudo code of the above approach is shown in Figure 3-19. In the implementation, we first find the horizontal part of the individual partitioning scheme and then using the horizontal partitioning position to complete the individual partitioning scheme. The details are given as follows: In line 2, we first evaluate the WSE of the whole matrix which treats all the rows and columns as in the same partition, i.e. no partition in the matrix at all. From line 3 to line 12, we find the horizontal part of the partitioning scheme by equation(3.21). From line 13 to 18, we

find out the vertical part of the partitioning scheme in the similar way by equation(3.20). In the equation, β is set as the vertical part of the partitioning scheme that has been found by equation(3.21). Finally, we compare the result in a partitioning case with the original matrix (the matrix before the split). If the partitioning case would result better solution, we further divide the left-top and right-bottom sub-matrices (refer to Figure 3-15, they corresponds to region A and D of the matrix respectively) in the current considering matrix, or otherwise halt the further division. The algorithm would be run recursively until there was no more individual partitioning scheme needed.

To calculate the WSE of the sub-matrix, we can use efficient way which has been used in the dynamic programming approach. Refer to the pseudo code in Figure 3-19, we can see that the algorithm searches for all the possible vertical (or horizontal) bi-sections and evaluates the corresponding WSEs, which is scanned from left to right, or vice versa. In fact, when the algorithm is searching the best cutting position, it has to evaluate the sub-matrices in which most of their areas in the sub-matrices have been evaluated in the past iterations. As a result, a modified version is proposed to reduce the duplicated evaluation of the certain area of the data matrix. This can be done by incremental sum up of the scanned region of the data matrix. For the sake of simpler illustration, we explain how it works in the vertical bi-section only. First, we have to initialize two array with 1 x dimension of column order (for horizontal bi-section case, the width would be dimension of row order), say colSSE_0[1..col] and colSSE_1[1..col]. Since the horizontal bisection point is fixed, we formulate colSSE_0 as:

$$colSSE\_0[i] = colSSE\_0[i-1] + \sum_{j=1}^{\beta} |a_{ij} - 1| + \sum_{j=\beta}^{n} |a_{ij} - 0| \quad (3.22)$$

colSSE_1 could also be formulated similarly:

$$colSSE\_1[i] = colSSE\_1[i-1] + \sum_{j=1}^{\beta} |a_{ij} - 0| + \sum_{j=\beta}^{n} |a_{ij} - 1| \quad (3.23)$$

Since the sub-blocks with positive correspondence to be identified are along the diagonal line of the data matrix, by scanning through all the possible combination of colSSE_0 and colSSE_1, we could also find the feasible vertical bisection point as the original formula. The revised formula would be:

$$vert.cut = \arg\min_{c \in COL} \{colSSE\_0[c] + colSSE\_1[c]\} \quad (3.24)$$

By this way, the algorithm could perform in a very efficient and it is proved to be efficient even in the large problem, which would be shown in the experiment part.

Compared with dynamic programming approach and heuristic partitioning approach, we would find that the result of the latter one is competitive with the former one, i.e. the WSE found is not as worse as the WSE found in the DP approach. In the experiment part, we would show out the comparison of the WSE found between the DP approach and the heuristic partitioning approach.

3.5 Experimental Study

In this section, the performance of BTACO clustering method is studied thoroughly. The effect of different conditions on the method is investigated. First, the comparison between the dynamic approach and heuristic partitioning approach is studied. Then, the different setting of the parameters in BTACO is investigated. The

study is divided into two parts, the study on the parameters for search direction and the one for reinforcement learning. The first part involves the study on the effect of δ and β to the result of the clustering method, while the second part involves the study on the effect of q_0, α as well as ρ (α and ρ would be considered to be the same parameter, as explained later) to the result.

The method is also compared with two clustering method: Genetic-algorithm based method and the average-link hierarchical clustering method. Since both methods could not guarantee the formation of a diagonal block structure in the solution matrix, an enumeration method is required to find out the most feasible set of sub-matrices.

3.5.1 Data set

In the following experiment, the data matrix for the vertical partitioning problem is used to analyze the performance of BTACO clustering method. To study the performance of BTACO clustering method on different conditions of the data matrices, a matrix generator is used to evaluate the robustness of the method. The matrix generator is random-based to generate different matrices with different parameters so that the robustness of the clustering method would be investigated. The matrix generator works as follow:

Parameters	Range
No. of row elements (row)	Integer
No. of column elements (col)	Integer
No. of Partitions (P)	Integer, <min(row,col)
Chance of swapping bit (S)	[0,0.5]

Figure 3-20 Parameters for matrix generator

In the matrix generator, there are four parameters to be set as seen in Figure 3-20. The first two parameters (*row* and *col*) control the dimensions of the data matrix, while P controls the number of partitions to be expected in the data matrix. S, which is defined as the chance of swapping bit, is a parameter for controlling the randomness of the data in the matrix. It can be regarded as the degree of noisy data in the matrix. In order to form different sizes of sub-matrices in the generated matrix, the generator divides both row and column order into P partitions by selecting P-1 cutting position by:

$$Cr_i = \frac{row}{P} + \frac{row}{P \times 3} \times Norm(1,0)$$

$$Cc_i = \frac{col}{P} + \frac{col}{P \times 3} \times Norm(1,0)$$

, where Ca_i and Ct_i are the cutting positions on the row and column order in the dimension respectively. Norm(1,0) is the random number generator which satisfies the standard normal distribution. The normal distribution is used on the determination of the cutting position to prevent the formation of unreasonable block structure on the matrix. The preset distribution could control the cutting position to be within [order/P ±order/P] with 99% confidence level, while there is 66.7% confidence level that the cutting position should be within [order/P ±order/3P]. In this way, different sizes of sub-matrices could be built and the dimensions of the sub-matrices should be with reasonable size. The diagonal block structure would be assigned to the partitioned matrix after the determination of the cutting positions. For the vertical partitioning problem, access frequencies are assigned to each transaction pattern. To simulate the effect on the access frequencies, weighting are then assigned to each column elements in the data matrix. The row and column order

in the matrix are then shuffled. Finally, to avoid solving a trivial problem, some noise data is introduced into the matrix. This might be done by applying some chances of swapping the bit of the elements in the two-dimensional matrix, e.g. changing the bit from 0 to 1 or from 1 to 0. The higher the chance of swapping the bit, the higher is the degree of noise data in the matrix and hence the problem could be more difficult to be solved.

3.5.2 Study on DP Approach and HP Approach

Dynamic programming approach may find the more accurate partitioning scheme for evaluation. However, its quadratic time complexity makes it impossible for applying to large data matrices. Hence, a heuristic partitioning approach is needed. In this section, we would compare the results of the two BTACO which uses different evaluation approaches.

Mat. ID	Matrix	Swap Deg.	ACS (DP)	ACS (HP)	gap (%)	DP_time	HP_time	Time_diff (in seconds)
1	20x15	-	**10.30**	**10.30**	0.00%	7.76	0.86	6.90
2	40x15	0.05	**12.54**	**12.54**	0.00%	20.94	10.20	10.74
3	40x15	0.05	**12.60**	**12.60**	0.00%	2.95	0.26	2.69
4	40x15	0.10	**26.26**	**26.26**	0.00%	36.23	0.90	35.33
5	40x15	0.10	**24.66**	24.68	0.08%	124.85	0.74	124.11
6	40x15	0.20	**51.00**	**51.00**	0.00%	184.87	23.30	161.57
7	40x15	0.20	**53.77**	54.33	1.04%	323.04	17.00	306.04
8	40x15	0.30	**79.66**	80.9	1.56%	262.31	21.01	241.3
9	40x15	0.30	**106.23**	106.43	0.19%	262.82	31.41	231.41
10	50x50	0.05	**60.58**	62.78	3.63%	360.74	201.14	159.60
11	50x50	0.05	**65.26**	**65.26**	0.00%	244.16	46.82	197.34
12	50x50	0.10	**133.35**	**133.35**	0.00%	2508.07	0.39	2507.68
13	50x50	0.10	139.46	**139.26**	-0.14%	4145.17	98.46	4046.71
14	50x50	0.20	**244.66**	257.18	5.12%	5114.60	760.17	4354.43
15	50x50	0.20	**236.58**	260.68	10.19%	1212.39	46.82	1165.57
16	50x50	0.30	**369.02**	375.76	1.83%	2352.28	50.82	2301.46
17	50x50	0.30	**319.76**	327.08	2.29%	663.59	36.21	627.38
18	50x200	0.05	**246.16**	**246.16**	0.00%	6761.67	38.43	6723.24
19	50x200	0.05	**261.42**	**261.42**	0.00%	1348.00	1.80	1346.20
20	50x200	0.10	**512.28**	516.80	0.88%	6619.26	135.52	6483.74
21	50x200	0.10	551.54	**548.82**	-0.49%	4593.15	1736.00	2857.15
22	50x200	0.20	**1261.22**	1405.00	11.40%	2789.60	18.02	2771.58
23	50x200	0.20	**1170.16**	1315.88	12.45%	2798.00	90.30	2707.70
24	50x200	0.30	**1614.90**	1640.26	1.57%	3529.40	2752.19	777.21
25	50x200	0.30	**1845.00**	1870.50	1.38%	8317.04	505.33	7811.71

(a) Random generated matrices with expected partitions = 4

Mat. ID	Matrix	Swap Deg.	ACS (DP)	ACS (HP)	gap (%)	DP_time	HP_time	Time_diff (in seconds)
26	50x50	0.05	**54.43**	**54.43**	0.00%	621.26	24.07	597.19
27	50x50	0.05	**61.14**	72.76	19.01%	5424.93	58.05	5366.88
28	50x50	0.1	**136.61**	150.61	10.25%	3662.75	144.41	3518.34
29	50x50	0.1	130.56	**128.36**	-1.69%	882.69	111.93	770.76
30	50x50	0.2	**267.88**	286.26	6.86%	3345.27	158.00	3187.27
31	50x50	0.2	**246.65**	263.86	6.98%	5784.71	65.26	5719.45
32	50x50	0.3	**450.42**	456.30	1.31%	2503.08	40.57	2462.51
33	50x50	0.3	**481.60**	493.50	2.47%	3021.77	29.89	2991.88
34	100x50	0.05	**172.82**	212.40	22.90%	6162.76	350.87	5811.89
35	100x50	0.05	**140.54**	145.06	3.22%	2141.30	266.41	1874.89
36	100x50	0.1	**259.12**	268.22	3.51%	2403.93	123.15	2280.78
37	100x50	0.1	**230.60**	243.10	5.42%	3081.03	255.56	2825.47
38	100x50	0.2	**544.67**	564.57	3.65%	5442.49	445.50	4996.99
39	100x50	0.2	**661.90**	681.70	2.99%	1805.27	208.42	1596.85
40	100x50	0.3	**639.90**	648.03	1.27%	408.85	298.87	109.98
41	100x50	0.3	**633.20**	652.28	3.01%	2418.83	46.39	2372.44

(b) Random generated matrices with expected partitions = 6

Figure 3-21 Comparison between DP approach and HP approach without learning

In the experiment for comparing DP approach and HP approach, we have generated several matrices with different sizes and different expected number of partitions. The information about the matrix generated is shown in Figure 3-21. To compare the performances of both approaches, we have tested both approaches with 5 trials for each matrix using the following parameters: No. of iteration = 500, $\alpha = \rho = 0.1$, $q_0 = 0.9$, $\delta = 0.2$ and $\beta = 3.0$. The reason for the choice of this configuration will be discussed in the next section. For the value of t_0, it is set as $(k \quad L_{kk})^{-1}$, where L_{kk} is the evaluation measurement from the first pass of the BTACO algorithm. k would be the order of row or column which depends on the order of tour the ants are moving

through. In the default setting, there would be (n+m) ants to be used in the algorithm, with each ant starting at different order (either row or column) position. n and m are the dimensions of the given data matrix. Since the computational time for the dynamic programming approach increases dramatically when the size of the matrix increases, we limit the time for evaluation to be within 2 hours. In the Figure 3-21, we have shown the results with minimum WSE found and their times to reach the best solution found. We may see that the difference of WSE found varies from 29.13% to -1.69%. In general, the dynamic programming approach could result a better matrix, no matter with the size of the matrices and the number of expected partitions. In fact, the ratio of difference is somewhat high which indicates that the dynamic programming approach is better than the heuristic partitioning approach in finding out a fit partitioning scheme. On the other hand, when considering with the time elapsed, for example, considering the 100 x 50 matrix with Matrix ID (34), dynamic programming approach costs over one and a half more hours to finish the BTACO, even though it have 22.9% of improvement. The improvement is not significant when the degree of noise is not great, for example considering the matrix with Matrix ID (26), both heuristic partitioning and dynamic programming approach results with the same WSE, but the latter approach costs 10 more minutes to get the same result. In the trade-off between accuracy and evaluation time, it is properly better to choose the heuristic partitioning approach as it is more efficient. The heuristic partitioning approach depends more highly on a formation of diagonal block structure in the data matrix, when the re-arranged data matrix is in better block structure, the heuristic partitioning approach should get similar result with the dynamic programming approach. Also, in the real case analysis, the experiment of the same setting would be run for several times and only the best solution is

recorded, the difference between the dynamic programming approach and the heuristic partitioning approach should be smaller.

Furthermore, in some cases, the results from the dynamic programming approach are not better than those from the heuristic partitioning approach (which is indicated by the negative ration in the comparison result, say -1.69%). This is due to the computational time restriction we have set for each trial of the experiment. Since the dynamic programming approach requires much more time for each iteration, the number of iteration could be reached is much smaller than that in the heuristic partitioning approach. Since there are few iterations in the ACO-algorithm to be run, the information learnt from the ants is not enough to get a fit partitioning scheme. As a result, the dynamic programming approach could not get a good result.

As the time elapsed for using dynamic programming approach is too large, in the following experiment, we would use the heuristic partitioning approach for the analysis.

3.5.3 Study on parameter settings

In this section, the study on the effect of the parameters on the performance of BTACO clustering method is investigated. As aforesaid, the study is divided into two parts; the first part involves the study on the parameters for the search direction, while the second part involves the study on the parameters for the reinforcement learning. For both parts, the same set of data matrices would be used for comparison. The visualizations of the matrices are shown in Figure 3-22, Figure 3-23 and Figure

3-24 (see Appendix I for detailed original matrix) while the information of the matrices are shown in Table 3-1. Since the data matrices are large, the raw data matrices are shown as color map, in which the darker the box in the matrix, the higher is the access frequencies associated. For the first matrix, it is the one proposed by Navathe et al for the case in vertical partitioning problem. For the second and the third matrices, they are formed by the matrix generator with the parameters shown in Table 3-1.

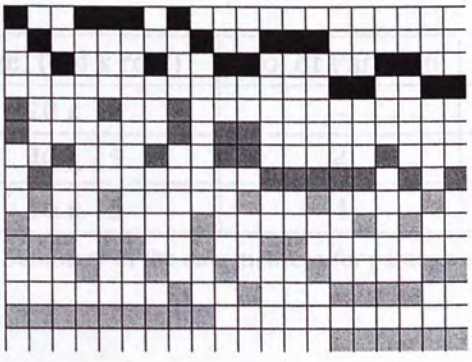


Figure 3-22 Data matrix 1 – Adapted from Navathe et al example

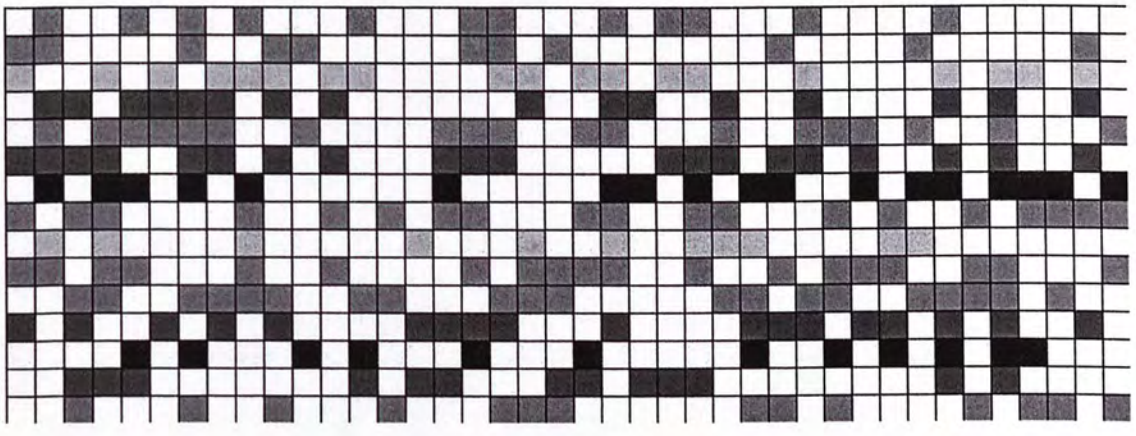


Figure 3-23 Data matrix 2

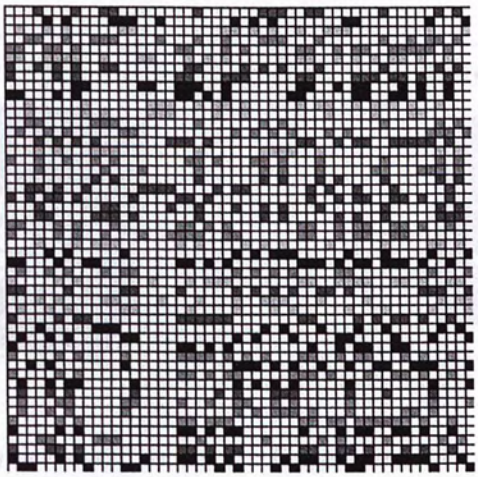


Figure 3-24 data matrix 3

Matrix set	Size: (col x row)	No. of Partition	Swap bit chance
1	20 x 15	-	-
2	40 x 15	4	0.3
3	50 x 15	4	0.1

Table 3-1 Summary of the data matrices for parameter settings

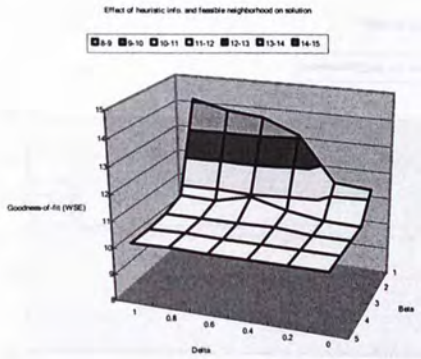


Figure 3-25 Effect of delta(δ) and beta(β) on the solution found in **data matrix 1** without learning effect

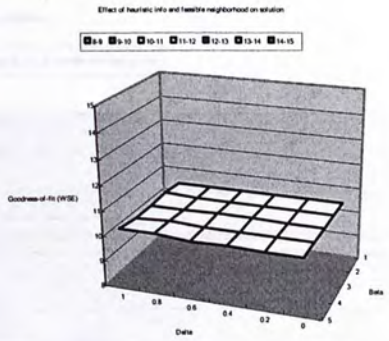


Figure 3-26 Effect of delta(δ) and beta(β) on the solution found in **data matrix 1** with learning effect

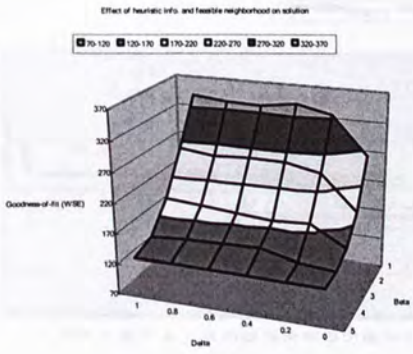


Figure 3-27 Effect of delta(δ) and beta(β) on the solution found in **data matrix 2** without learning effect

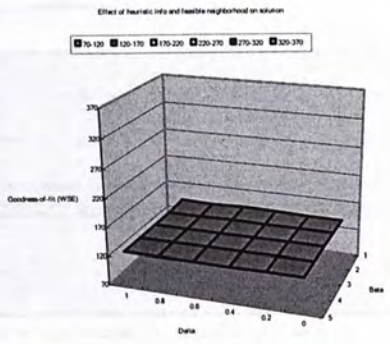


Figure 3-28 Effect of delta(δ) and beta(β) on the solution found in **data matrix 2** with learning effect

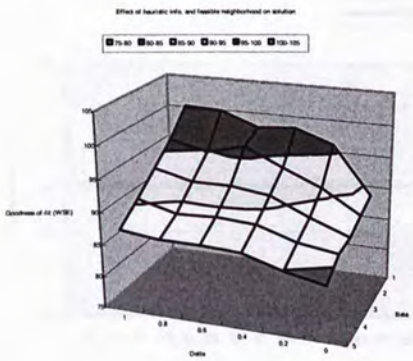


Figure 3-29 Effect of delta(δ) and beta(β) on the solution found in **data matrix 3** without learning effect

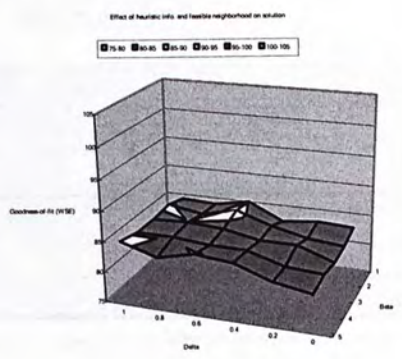


Figure 3-30 Effect of delta(δ) and beta(β) on the solution found in **data matrix 3** with learning effect

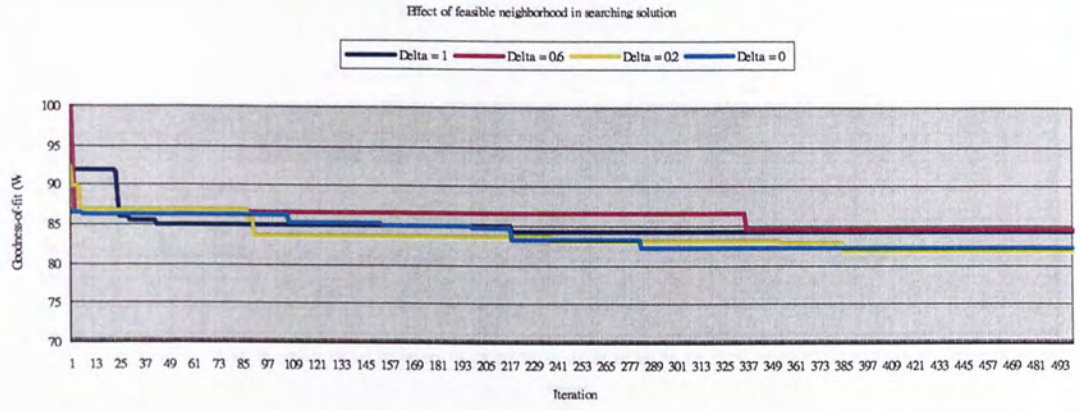


Figure 3-31 Improvement curve for $\beta = 1$ in **data matrix 2**

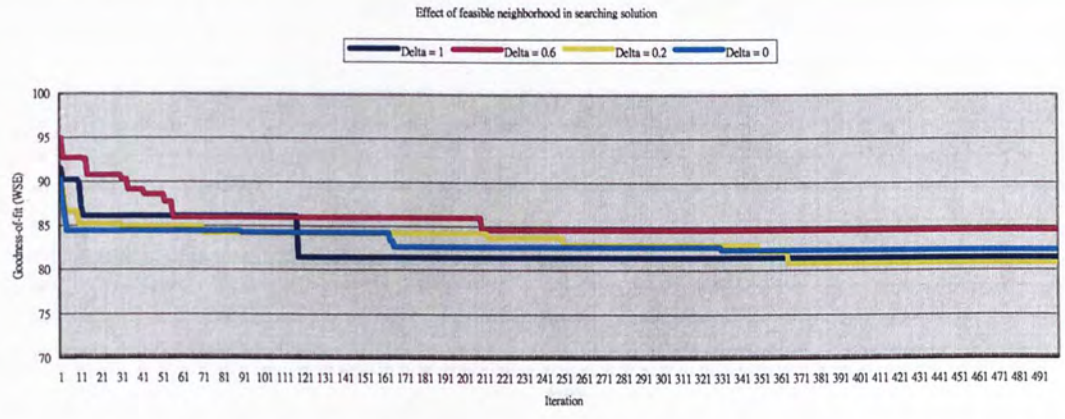


Figure 3-32 Improvement curve for $\beta = 3$ in **data matrix 2**

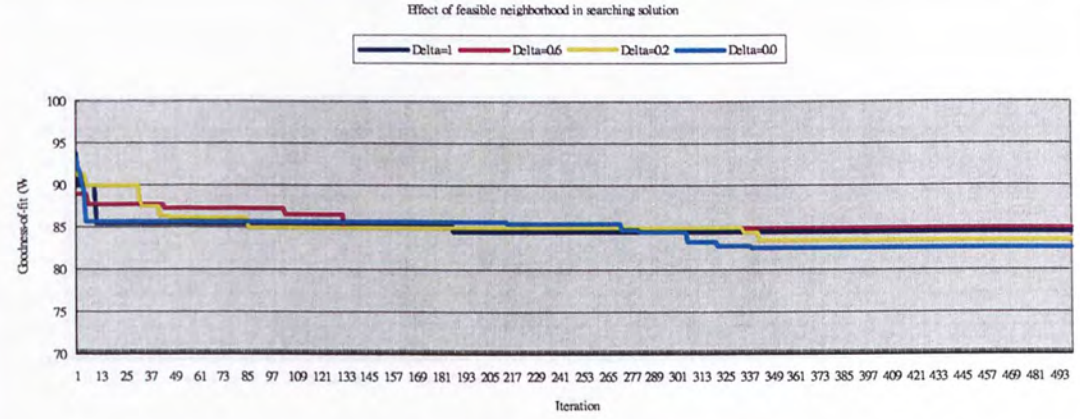


Figure 3-33 Improvement curve for $\beta = 5$ in **data matrix 2**

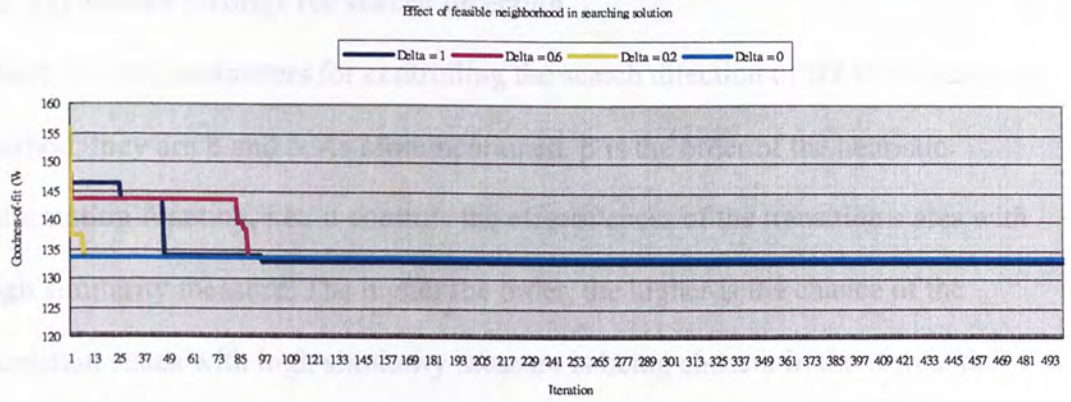


Figure 3-34 Improvement curve for $\beta = 1$ in **data matrix 3**

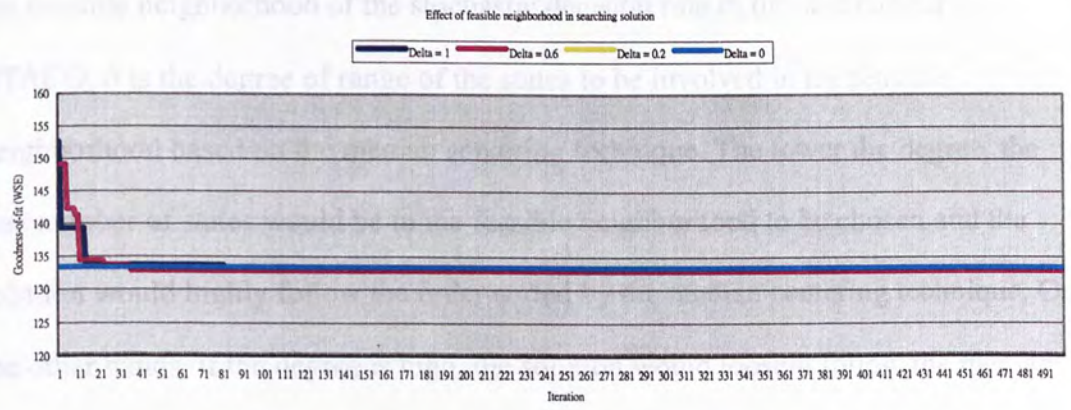


Figure 3-35 Improvement curve for $\beta = 3$ in **data matrix 3**

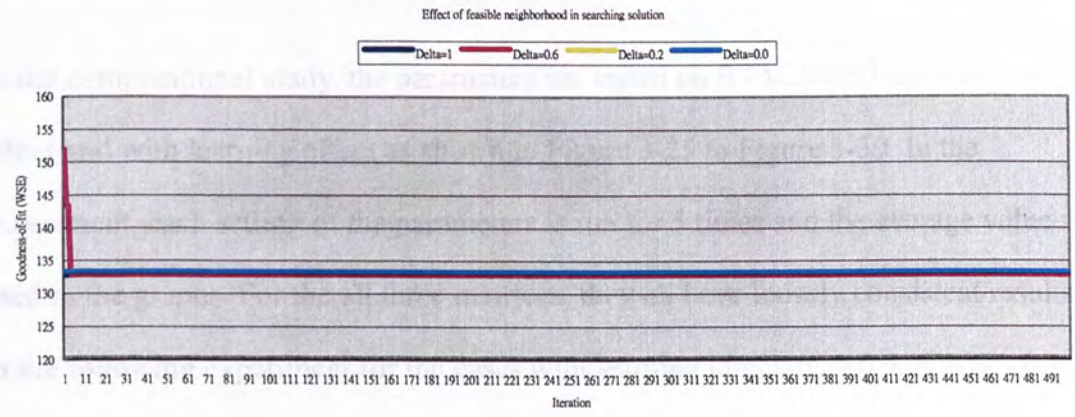


Figure 3-36 Improvement curve for $\beta = 5$ in **data matrix 3**

A. Parameter settings for search direction

There are two parameters for controlling the search direction of BTACO clustering method, they are β and δ. As aforementioned, β is the order of the heuristic information function, i.e., it controls the effectiveness of the transition states with high similarity measure. The higher the order, the higher is the chance of the transition states with high similarity measure is being chosen. In the words, the choices of the search are narrowed to be focus on the ones with high similarity measure. δ is another parameter for the control of search direction by constraining the feasible neighborhood of the stochastic decision rule in the second tour of BTACO. δ is the degree of range of the states to be involved in the feasible neighborhood based on the median centering technique. The lower the degree, the less number of states would be in the feasible neighborhood to be chosen and the solution would highly follow the order sorted by the median centering technique. On the other hands, if the degree is high, the solution would loosely follow the median centering-sorted order, but the possible permutation of the solution would be in larger number.

In the computational study, the parameters are tested on BTACO without learning effect and with learning effect as shown in Figure 3-25 to Figure 3-30. In the experiment, each setting of the parameters is run for 5 times and the average value is used in the graphs. For the all three matrices, they all have loosely consistent results. In the following experiment for the cases with learning effect, $q_0 = 0.9$, $\alpha = \rho = 0.1$. For the value of t_0, it is set as $(k \quad L_{kk})^{-1}$, where L_{kk} is the evaluation measurement from the first pass of the BTACO algorithm. k would be the order of row or column which depends on the order of tour the ants are moving through. In the default

setting, there would be (n+m) ants to be used in the algorithm, with each ant starting at different order (either row or column) position. n and m are the dimensions of the given data matrix. In the below experiments, we would have a 500 trails for BTACO to search for the solution.

When the learning effect is not possible, we see that the lower the beta, the higher is the weighed sum of error (WSE), i.e., the goodness-of-fit is lower in such cases. The higher power order of the distance measure may cause the algorithm to search the states with longer distance measure with lower chance. This means the heuristic information which is represented as the similarity measure is useful for finding the solution, as the states with shorter distance could indeed form the solution with high value of WSE. The low power order of heuristic information cause the algorithm to search more different states and so it could not find out the desirable solution within the limited number of iteration.

δ is the parameter for controlling the bound of median value in which the states chosen should satisfy. Considering the effect of δ, we could see that the higher the value of δ, the higher is the resulted weighed sum or error (WSE) in general. This means that the median centering technique is useful in finding out the desirable solution for the problem. As for the high value of δ, the states in the feasible neighborhood are not necessary to be within the same value of median, and the algorithm would prefer to choose the states with higher similarity by lower distance measure. As aforementioned, the similarity distance measure is effective at finding out the groups of similar objects, but is not necessary to be reliable in finding out the most desirable groups, especially when there is much noise data within the data

matrix. As a result, the large feasible neighborhood diverge the search direction of the algorithm and so the solution is worse when δ is large.

The graphs in Figure 3-25 to Figure 3-30 show the result of the algorithm when learning effect is activated. From all the graphs, we could see that the solutions are roughly the same no matter what the values in the parameters. This nice result may be contributed by the reinforcement learning in BTACO. As the feedback mechanism could "teach" the algorithm that which transition from one state to another state is worse and which one is better. Iteratively, the algorithm learns through the solution and hence this implies that the learning effect is very important in BTACO clustering method.

Although the results attained are roughly the same, the parameters affect the progression in getting the solution. In Figure 3-31 to Figure 3-36, the improvement curve for different values of δ and β is shown. An improvement curve keeps track of the best solution found throughout all the trials of the experiment. For each parameter settings, we used the data from the trial attaining the median results among all the trials for comparison. For Figure 3-31 to Figure 3-33, they are the improvement curves for matrix 2 as described previously. We see that although they all result in roughly the same weighed sum of error, the number of iteration required differs from different setting of the parameters. For the high value of β, the best solution attains at a lower WSE compared with the one with low value of β. As discussed before, the high power order of the heuristic information for the stochastic decision rule could result in the algorithm concentrating on the transition of states with low distance measure. As a result, the best solution would start with a lower

WSE. The same phenomenon occurs in the case of δ. The low value of δ could result in the narrower search direction which is appropriate at finding out the desired solution and hence the best solution starts with lower WSE at the earlier stage of the improvement curve.

Even though the narrower the search direction, the better the solution is in the case without the learning effect. This is not absolutely true for the case with learning effect. Compared with the Figure 3-31 to Figure 3-33 again, we see that the ratio of improvement in Figure 3-33 is not as great as those in Figure 3-31 and Figure 3-32, also the final solutions in the first two graphs are better than that in the third graph. Although the difference between the value of the goodness-of-fit is not great and the values vary from one trail to another, in general, the higher power order of heuristic information, the higher is the WSE attained and hence the result would be less desirable than those got with lower power order of heuristic information. This is because the high order had narrowed the search direction and hence less possible solutions had been explored. With the same sense, the lower the value of δ, the worse the result are gotten. This is because although the median centering technique could help in the formation of appropriate diagonal block structure, it again narrowed the search direction and hence less number of possible solutions is being explored.

The similar results may also be gotten in Figure 3-37 and Figure 3-38, except that the curves level off at the earlier stage within 500 iterations, as the matrix 3 is generated with 0.1 chance of swapping bit, it is more trivial than that in matrix 2. However, it is interesting to see that although the best results are roughly the same,

the methods with lower β and higher δ could have a higher chance in getting a little improvement in the solution. This means that the high degree of exploration could help the algorithm in getting away from the local optimal result.

In the conclusion, the value of β and δ should not be set to extreme high and low value to get a balance between exploration and exploitation. The value of β = 3 and δ = 0.2 is suggested throughout the following experiment.

B. Parameter for reinforcement learning

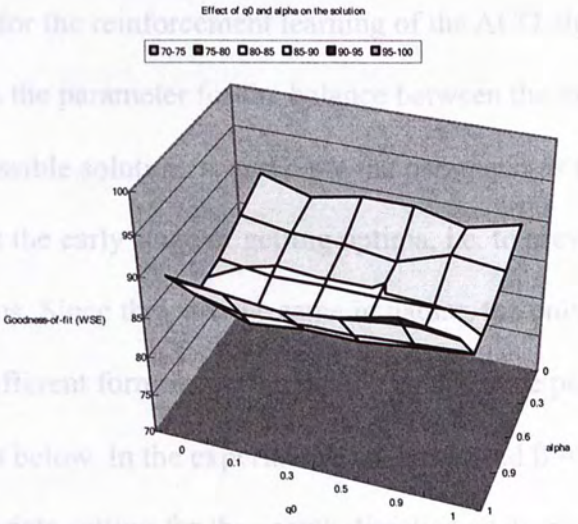


Figure 3-37 Effect of learning parameters on the solution found in data matrix 2

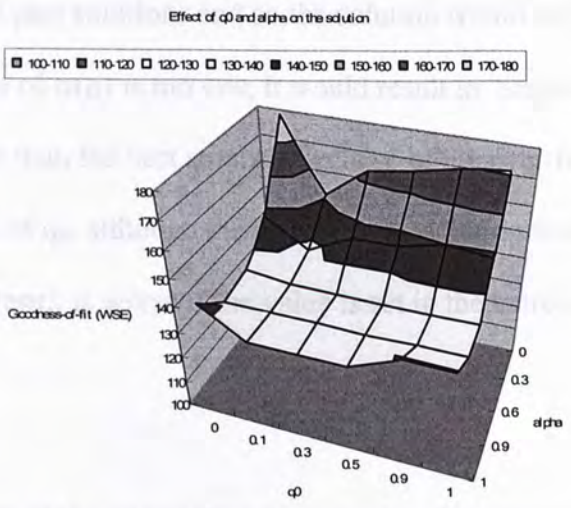


Figure 3-38 Effect of learning parameters on the solution found in data matrix 3

Since BTACO is ACO-based algorithm, it has the ability of learning from the solution attained by each agent and so as to improve the performance. The q_0, α and ρ are the parameters for the reinforcement learning of the ACO algorithm. As aforementioned, q_0 is the parameter for the balance between the exploitation and exploration of the possible solution. α and ρ are the parameter as the decay coefficient to prevent the early stage of getting optima, i.e. to prevent from getting local optimal solutions. Since they are the same in nature, the only difference is that they are applied in different formulas, we treated it as the same parameter in the performance analysis below. In the experiment, we have used $\beta = 3.0$ and $\delta = 0.2$ which are the appropriate setting for the search direction as discussed above. We run the trails for 5 times and the average WSE values are used for comparison. In each set of experiment, 500 iterations are run for getting the best global solution. From the result got from data matrix 2 and 3, we could see that the appropriate setting of α (ρ) and q_0 is roughly the same as that suggested by Dorigo et al in [34]. In fact, the high value of $\alpha(\rho)$ would cause the algorithm could not learn from last solution but

not the whole series of past solutions and so the solution would not be good. On the other hand, if the value of $\alpha(\rho)$ is too low, it would result in "trapping" into the local optimal solution rather than the best solution because of the over-fitting problem. Considering the effect of q_0, although the effect on the solution found is not obvious, we could still see the result is worse if the value is set to the extreme value, i.e. 0 or 1.

In general, if the values of the parameters are not set to the extreme values, the BTACO could work well in getting the desirable solution. And from the performance analysis, we suggest to use $\alpha = \rho = 0.1$, $q_0 = 0.9$, $\delta = 0.2$ and $\beta = 3.0$ in the following experiment.

3.5.4 Comparison with GA-based & hierarchical clustering methods

In the previous sections, the performance of the BTACO on different factors has been discussed. In this section, the comparison of the performance with two different clustering methods is shown. The GA-based clustering algorithm and the average -linkage hierarchical clustering method are used for comparison.

The GA-based clustering algorithm proposed by Cheng et al[23] is used for comparison with our algorithm. In the GA-based algorithm, it clusters the data matrix by re-arranging the data matrix so that the similar rows and columns are grouped together. This may be done by solving the associated traveling salesman problems defined by dimension of the data matrix separately. In the comparison, the

enhanced edge recombination (EER) crossover is used as the crossover method in the genetic algorithm. In fact, there are three parameters to be set in the GA-based algorithm; they are the population size, termination criteria and the selective pressure. In our experiment, the population size is set as 2000, the selective pressure as 1.3 and the termination criteria as "the global best solution = 99.85% of the average value in the population".

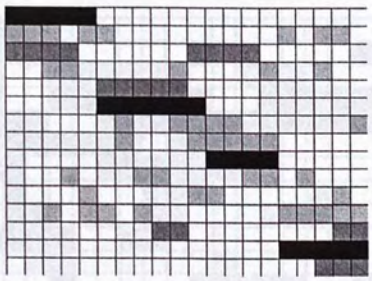


Figure 3-39 Re-arranged **data matrix 1** by using BTACO

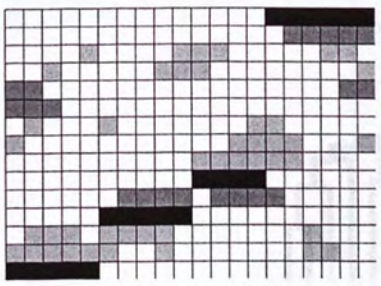


Figure 3-40 Re-arranged **data matrix 1** by using GA

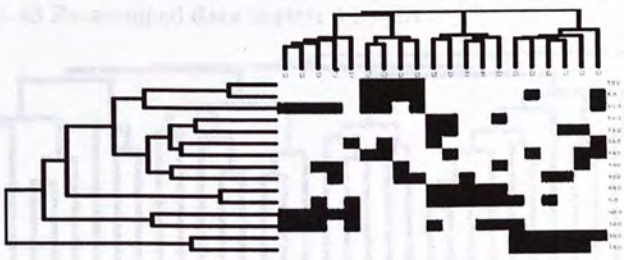


Figure 3-41 Re-arranged **data matrix 1** by using hierarchical clustering

Figure 3-39 to Figure 3-41Re-arranged **data matrix 1** for BTACO, GA-based algor. and hierarchical clustering respectively

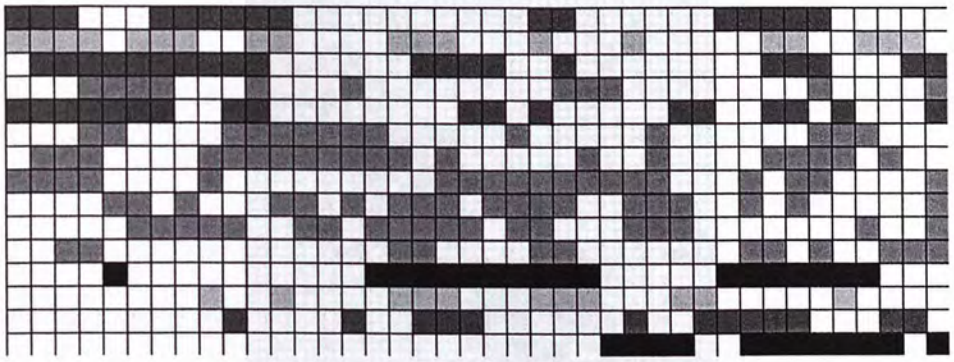


Figure 3-42 Re-arranged **data matrix 2** by BTACO

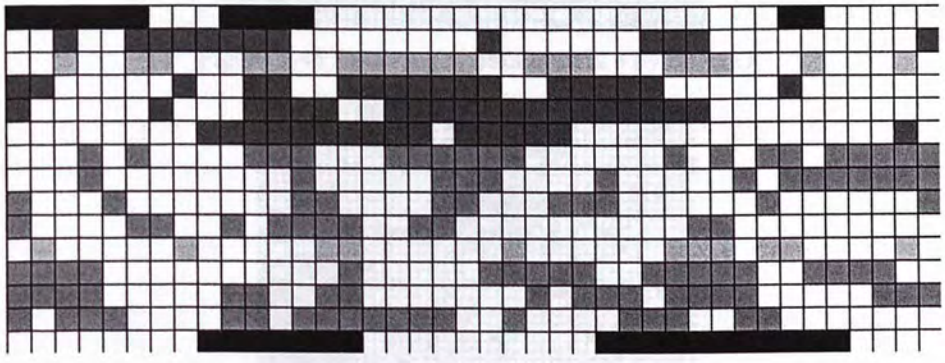


Figure 3-43 Re-arranged **data matrix 2** by GA

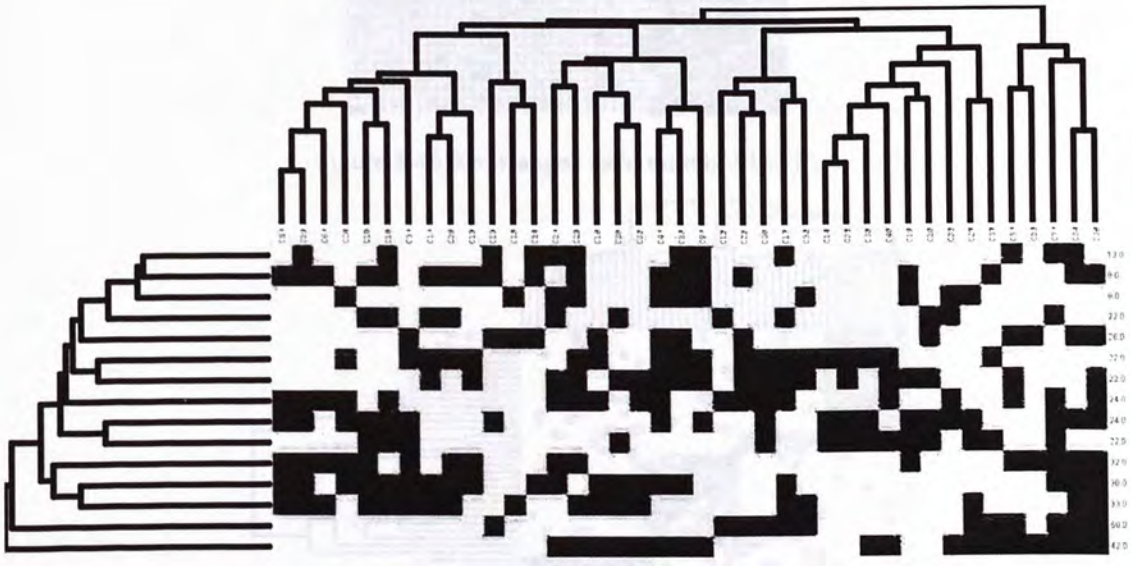


Figure 3-44 Re-arranged **data matrix 2** by hierarchical clustering

Figure 3-42 to Figure 3-44 Re-arranged **data matrix 2** for BTACO, GA-based algor. and hierarchical clustering respectively

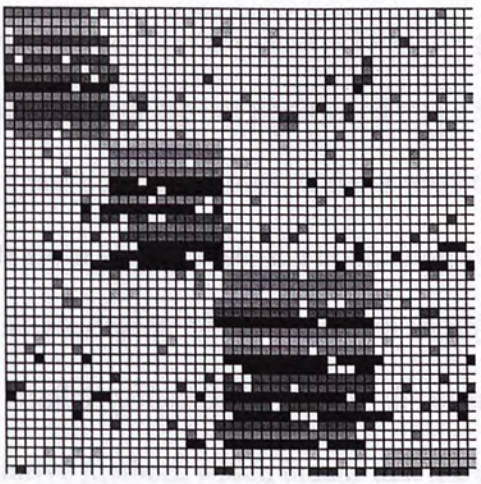


Figure 3-45 Re-arranged **data matrix 3** by BTACO

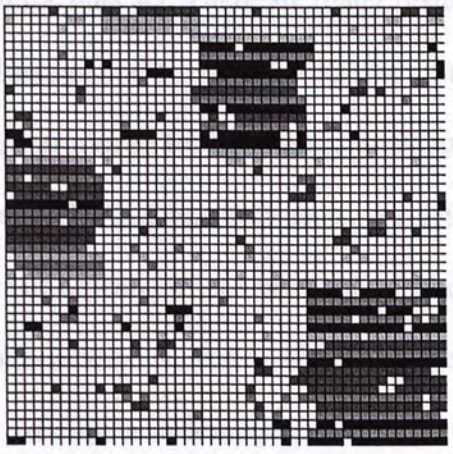


Figure 3-46 Re-arranged **data matrix 3** by GA

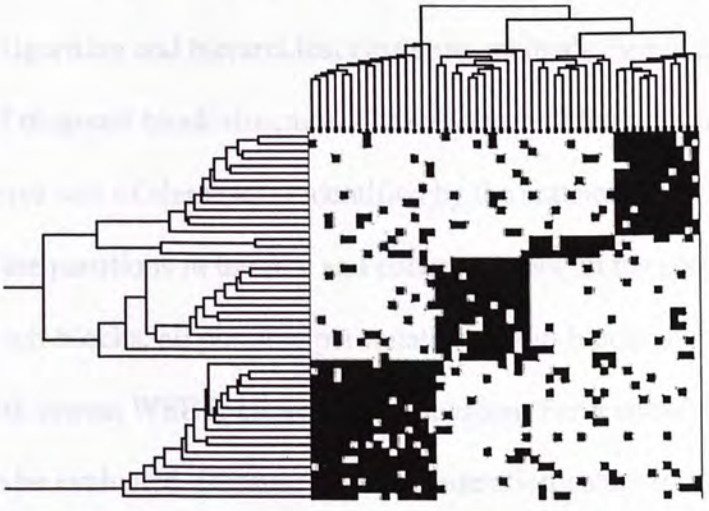


Figure 3-47 Re-arranged **data matrix 3** by hierarchical clustering

Figure 3-45 to Figure 3-47 Re-arranged **data matrix 3** for BTACO, GA-based algor. and hierarchical clustering respectively

The hierarchical clustering methods have been developed for years[55]. It is a clustering method in which the data are not partitioned into a set of clusters as the result. Instead, a series of partitions are shown, which shows the partitioning relationship of the objects from single clustering containing all the objects to n clusters each containing single object. To determine which objects are similar in each step of the series, a similarity distance is used as the measurement. Hierarchical clustering is an agglomerative method, which means that the cluster is formed by the fusion of the sub-cluster in the data set. There are many fusion methods, such as single-linkage, complete-linkage and average-linkage, etc. In our experiment, the average-linkage method is introduced for the hierarchical clustering method. In the definition of average-linkage, the distance between two clusters is as the average of distances between all pairs of objects, where each pair is made up of one object from the two groups. The average-linkage is used as we expect the average distance could reduce the effect of noise data in the data matrix.

The GA-based algorithm and hierarchical clustering methods cannot guarantee in the formation of diagonal block structure. The collection of the sub-blocks with mutually exclusive sets of elements is identified by the enumeration of the possible permutation of the partitions in the row and column orders. In the process of identifying the sub-blocks, all possible permutation of sub-blocks are identified and only the one with lowest WSE is chosen as the solution. For n clusters, there would be $n!$ choices to be evaluated. Because of the enumeration nature of the identification, for the matrix generator, the number of partition is set to be small, say 4 or 6, in all the cases. And in the evaluation, we could only check the number of clusters to be up to 12.

Mat. ID	Matrix	Swap Deg.	ACS (DP)	ACS (HP)	GA	HC
1	20x15	-	**10.30**	**10.30**	15.00	11.30
2	40x15	0.05	**12.30**	**12.30**	14.62	**12.30**
3	40x15	0.05	**12.60**	**12.60**	19.95	19.95
4	40x15	0.1	**26.26**	**26.26**	56.26	36.82
5	40x15	0.1	**24.66**	24.68	40.64	32.30
6	40x15	0.2	**51.00**	**51.00**	93.20	74.29
7	40x15	0.2	**53.77**	54.33	72.86	65.67
8	40x15	0.3	**79.66**	80.90	96.24	95.62
9	40x15	0.3	**106.23**	106.43	117.36	129.06
10	50x50	0.05	**60.58**	62.78	228.04	88.36
11	50x50	0.05	**65.26**	**65.26**	224.36	**65.26**
12	50x50	0.1	**133.35**	**133.35**	225.32	150.64
13	50x50	0.1	139.46	**139.26**	192.57	158.06
14	50x50	0.2	**244.66**	257.18	319.84	291.30
15	50x50	0.2	**236.58**	260.68	350.50	370.74
16	50x50	0.3	**369.02**	375.76	401.53	399.12
17	50x50	0.3	**319.76**	327.08	409.30	408.36
18	50x200	0.05	**246.16**	**246.16**	789.44	246.16
19	50x200	0.05	**261.42**	**261.42**	748.20	261.42
20	50x200	0.1	**512.28**	516.80	842.36	549.12
21	50x200	0.1	551.54	**548.82**	647.18	520.66
22	50x200	0.2	**1261.22**	1405.00	1566.16	1463.52
23	50x200	0.2	**1170.16**	1315.88	1603.54	1637.82
24	50x200	0.3	**1614.90**	1640.26	1946.14	1717.22
25	50x200	0.3	**1845.00**	1870.50	1950.50	1945.94

(a) Random generated matrices with expected partitions = 4

Mat. ID	Matrix	Swap Deg.	ACS (DP)	ACS (HP)	GA	HC
26	50x50	0.05	**54.43**	**54.43**	119.61	68.45
27	50x50	0.05	**61.14**	72.76	101.54	83.52
28	50x50	0.1	**136.61**	150.61	197.33	186.31
29	50x50	0.1	130.56	**128.36**	248.84	186.30
30	50x50	0.2	**267.88**	286.26	304.86	296.72
31	50x50	0.2	**246.65**	263.86	284.88	284.08
32	50x50	0.3	**450.42**	456.30	508.06	504.90
33	50x50	0.3	**481.60**	493.50	537.82	528.34
34	100x50	0.05	**172.82**	212.40	262.40	**172.82**
35	100x50	0.05	**140.54**	145.06	234.30	148.82
36	100x50	0.1	**259.12**	268.22	439.74	301.54
37	100x50	0.1	**230.60**	243.10	336.36	269.68
38	100x50	0.2	**544.67**	564.57	605.37	593.98
39	100x50	0.2	**661.90**	681.70	789.38	748.18
40	100x50	0.3	**639.90**	648.03	700.25	694.17
41	100x50	0.3	**633.20**	652.28	702.71	682.76

(b) Random generated matrices with expected partitions = 6

Table 3-2 Performance comparisons among three different clustering methods

In Figure 3-39 to Figure 3-47, the visual results of the re-arranged data matrices by GA-based algorithm, the hierarchical clustering method and BTACO are shown. In the figures, we have only showed the color-mapped version of the re-arranged data matrix without partitioning scheme for clearer visualization (We have also shown the detailed re-arranged matrices with the partitioning scheme in the Appendix I for better understanding of the results generated by all the three methods). In data matrix 1, we see that BTACO could result in the nice diagonal block structure. For the GA-based algorithm, it may not result in the block structure even though the

same distance measure as BTACO is used. Also, although some sub-blocks are formed, it is scattered around the data matrix and is difficult to find out the relationship between the row and column orders. In the case of hierarchical clustering method, no sub-blocks could be traced out at all. However, when compared with the WSEs computed, GA-method got the worst result even though it has a rough diagonal block structure. The only reason for such adverse situation is that partitioning the matrix with largest distance measure may not be appropriate in the two-dimensional case, especially for identifying the horizontal cutting position. Refer to the detailed partitioning result in Appendix I (g), the partitioning scheme assigned the clusters without covering as much "1" as possible and so the WSE value got is somewhat high (refer to row with Matrix ID(1) in Table 3-2). This adverse result happens as the intuitive idea that proposed in Genetic Algorithm approach cannot be applied to the row data. In fact, solving associated TSP using distance measure is one of the variant of the bond energy approach [7, 82], in which the solution would be equivalent to maximizing the objective of the bond energy expression. However, the distance measure between the pair of row (column) does not give out any information about the cluster position. There is no explicit relationship between the cluster position and the distance. The high distance measure does not guarantee that the certain position is a good partitioning point. That's why Genetic Algorithm approach results in poor WSE even though the re-arranged matrix having clear structure of sub-blocks allocated around.

In data matrix 2, although all three matrices could not be re-arranged to be in diagonal block structure at all, BTACO clustering method try to organize the objects with high access frequencies to be together and less frequent as the outliers. In this

way, as aforesaid in the motivation part, this is a way to minimize the errors and outliers in the solution and so the high goodness-of-fit would be resulted.

In data matrix 3, all the three matrices could be re-arranged to be in diagonal block structure, in which the sub-blocks arranged in GA-based are not along the diagonal. In general, the set of sub-blocks identified are with similar shape and size among the data matrix, which could be verified by inspection easily. But when considering the partitioning scheme, Genetic Algorithm results in poor WSE (refer to row with Matrix ID (12) in Table 3-2) even though all of the re-arranged matrix could result in similar partitioning scheme by inspection. The reason is again due to the policy of partitioning the matrix at the point having high distance measure, which is not appropriate for row partitioning. In this case, the partitioning scheme proposed by this policy is totally unacceptable as the clusters are assigned on partial area of the sub-blocks. The situation is worse in the matrix 3 as the variation of the weightings among the row data is great. The pair of rows with similar 0-1 patterns but with higher difference in weightings would have higher distance than the one with totally different 0-1 patterns but with the same weightings.

In the table in Table 3-2, the goodness-of-fit measurement of the solution for different conditions of the data matrix are shown, in which the data matrix 1, 2 and 3 are also included. For the heuristic search method, i.e. the BTACO and GA clustering method, the measurement is resulted from 5 trials with the best result is recorded. From the table, we can see that BTACO, no matter using the dynamic programming or the heuristic partitioning approach, results in the better performance at all. Hierarchical clustering could sometimes result in the same performance with

BTACO when there is less noisy data implied in the data matrix. This can show that BTACO could have a higher tolerance in treating the noisy data. In the GA-approach, the inappropriate partitioning policy results in the poor performance. In fact, the policy of partitioning at the pair of high distance data may be only applicable in the column only [23]. In fact, as discussed before, there is no explicit correlation between the distance measure and the partitioning position in the data.

3.6 Chapter conclusion

In this chapter, we have introduced the Ant Colony Optimization algorithm-based block clustering, called Bi-Tour Ant Colony Optimization (BTACO) clustering method. In fact, we have introduced some ideas to the ACO-based algorithm so that the clustering method can perform the block clustering well. We have introduced the idea of using two tours in the algorithm for re-arranging the data matrix into diagonal block structure for better block identification. Also, two block identification approaches are proposed in which they differ from the time complexity and the accuracy of getting the best partitioning scheme. Dynamic programming approach suffers from quadratic time complexity but it could get a better partitioning scheme. On the other hand, heuristic partitioning approach can effectively get the partitioning scheme in linear time complexity but the poor WSE value got indicated that heuristic partitioning approach cannot get a solution which is as good as the one DP got. In overall, since the dynamic programming approach requires too much time in getting the partitioning scheme for the data matrix especially when the size of the matrix is large, heuristic partitioning approach would be better for our block clustering model.

In the computational study, the BTACO clustering method could perform the non-overlapping bi-clustering well compared with other clustering methods. It can directly identify the sub-blocks and hence the correspondence relationships among the rows and columns through the diagonal block structure. In some cases, although it is not possible to re-arrange the data matrix into diagonal block structure, it could still minimize the errors (outliers and errors inside sub-blocks) well compared with other methods. The use of BTACO clustering method could reduce the degree of replication of data in the design of the database system by reducing the outliers (which could form the inter-submatrix attribute). The ability of direct reveal of the correspondence of the row and column elements could also help in the cluster analysis. The diagonal clustering can visualize the inter-submatrix attributes as the outliers in the re-arranged data matrix which could be easier to be identified. In the vertical partitioning problem, after the identification of the inter-submatrix attributes, we can in fact deal with the outliers (inter-submatrix attribute) by either two ways[23]:

1. Duplicate the inter-submatrix attributes into all of the identified submatrices.
2. Create an additional submatrix attributes into all of the identified submatrices.

Since the strategy of dealing with inter-submatrix attributes is not in our scope, we have not discussed it in our work. In general, identification of the sub-blocks in the associated data matrix for specific working profile is in vital importance in the database system design.

In this next chapter, we will show how the BTACO clustering method to be applied

in clustering the XML documents in the database system so as to improve the efficiency of query processing.

Chapter 4 Application of BTACO-based clustering in XML database system

4.1 Introduction

In the last chapter, we have shown the use of BTACO-based clustering method to perform block clustering. In this chapter, we focus on the use of BTACO-based block clustering in a collection of XML documents in the XML database system so as to identify the schematic information as well as to improve the performance of the XML database system.

Extensible Markup Language (XML) plays an important role in defining a standard to represent data over the World Wide Web [90]. There is a rich XML literature in the use of XML. In this paper, we would like to concentrate only on clustering a collection of XML documents. This is useful in the identifying XML schemas and improving the performance of XML database systems. There are two kinds of clustering XML documents, they are clustering by structure and clustering by user transaction pattern.

The XML documents with similar structure could be clustered together to get a

schemas which are used to define the structure, content and semantics of the XML documents. The query engine in the XML database can then use the schematic information of the collection of XML documents for avoiding the irrelevant data from processing. Hence, the efficiency of the XML database system can be improved. However, clustering by structure requires specific similarity measurement as structural information is stored inside the XML documents. There are literatures discussing about the measurement of the structural similarity of the XML documents and we will discuss in detail later.

In a relational database management system, the vertical partitioning could be performed when the transaction pattern is available, in which the attributes which are usually accessed together would be grouped together. In this way, the disk accessed will be minimized and so the performance of the database system would be improved. Similarly, if the access pattern of a collection of XML documents is available, these XML documents may be clustered for better database performance. In the relational database system design, we cluster the database by partitioning the set of attributes involved. On the other hand, in the XML database system design, we will cluster the database by partitioning the collection of XML documents involved.

In the following, we will first briefly overview the techniques used in improving relational database design. Then we will discuss how the clustering of XML documents helps in organizing and improving an XML database system, based on the fact that clustering XML documents by structure and by access pattern could be analogous to the normalization and vertical partitioning of a relational model

respectively. Third, we will introduce the use of BTACO-based method for clustering the XML documents. Finally, we will show the computational study which shows the performance of BTACO-based clustering in clustering XML documents.

4.2 Overview of normalization and vertical partitioning in relational DB design

In the last section, we have mentioned that clustering XML documents by structure and by user access pattern is in fact analogous to the normalization and vertical partitioning of a relational database system. In order to have a better understanding in the analogism between the two different database models (i.e. relational data model and XML data model), we would like to have an overview of the two important relational database design issues: Normalization and vertical partitioning of relational database system in which their ideas could be applied to the XML data model. In the following, we will first briefly discuss about the normalization of the relational data model. Then, we will introduce the vertical partitioning and its practical methods proposed in literatures.

4.2.1 Normalization of relational models in database design

For a relational database model, one may use the database design techniques such as

the entity-relationship model for producing a good database design. However, there are some undesirable features, especially when the scope of the database design is large and complex. In order to prevent the database design from having undesirable features, such as repetition of information, update anomalies, insertion anomalies and deletion anomalies, it is necessary to perform normalization[95, 102]. The normalization process validates as well as modifies (e.g. by decomposition) the schemas of the database system so that the undesirable features could be eliminated. Almost at the same time when the relational database model was proposed by Codd[28], the concept of a normal form was also introduced for resulting in a good database design. Normalization of database design is to eliminate the undesirable features such as data redundancy and update anomalies in the database system. However, it may not improve the performance of the database system; rather, it requires additional joins for processing a query. To improve the performance of querying the database, it is the job of vertical partitioning in the design is needed[88].

In order to accomplish the prevention of anomalies, several stages of the normalization process are performed. In general, they include first normal form (1NF), second normal form (2NF), third normal form (3NF)[26], Boyce-Codd normal form (BCNF)[27], fourth normal form (4NF)[39] and fifth normal form (5NF)[40]. In the following, we would discuss about the properties of each normal form briefly.

A relation is in first normal form when it does not contain multi-valued or composite attributes[26, 28]. This satisfies the properties of relational theory. For a relation in

second normal form, it requires that all attributes that do not belong to candidate key should be fully dependent on each candidate key, in which the dependent relationship is determined by the functional dependencies of the attributes. For the relation in third normal form, it requires that the relation is in 2NF and all the non-key attributes in the relation are non-transitively dependent on each candidate key of the relation. For most cases, 3NF is adequate for most relational database design. However, anomalies may likely exist in some cases and necessitates in the use of Boyce-Codd normal form. BCNF deals with the anomalies due to the dependencies within the candidate keys. For a relation in BCNF, whenever the set of attributes A is dependent on set of attributes X and A is not in X, X is the candidate key for the relation. By the implementation of BCNF, all the anomalies and repetition of information should be removed. For the relation with multi-valued attributes, fourth normal form is required for the normalization. For a relation in 4NF, whenever there is a multi-valued dependency in which the set of attributes Y is dependent on set of attributes X, then either the dependency is trivial or X is the candidate key for the relation. By the implementation of 4NF, both the problem of data redundancy and anomalies will be eliminated. In order to perform lossless join decomposition in the relation, the fifth normal form, also known as project-join normal form (PJNF) will be introduced. In 5NF, it deals with the join-dependencies which could be regarded as the generalization of the multi-valued dependencies. A relation R satisfies the join-dependency if and only if the relation is equal to the join of the subset of the set of attributes in relation R. By the definition of join-dependency, we define the fifth normal as follow: For a relation in 5NF, for all the join dependencies, they should be the trivial join dependencies (i.e. one of the subset is the relation itself), or every subset is a candidate key for the relation or

both properties satisfied.

In the above, we have discussed the 5 typical normal forms for the relational database model. In the practical cases, 3NF or BCNF are adequate as they remove the anomalies discussed before in most common situations.

4.2.2 Vertical partitioning in database design

Vertical partitioning in RDBMS design is the process of assigning a logical object (relation) from the logical schema of the database to several physical objects (files) in a stored database[88]. The objective is to minimize the number of page accesses by creating smaller fragments in order to satisfy user queries[23]. In fact, the VP problem is a combinatorial problem. As pointed out by Navathe et al.[88], for a relation having n non-primary key attributes, the number of possible fragments to consider would equal the nth Bell number, in which $B(n) \approx n^n$.

There have been many approaches proposed for the vertical partitioning problem. Hoffer[60] proposed the binary nonlinear integer programming model for solving the problem. The objective of the problem is to minimize the storage, retrieval as well as the update costs subject to capacity constraints on the database sub-files. Bond energy algorithm (BEA) is proposed to result in an approximate solution for the objective function. Navathe et al. [88] extended the work by Hoffer. The bond energy algorithm is introduced to partition the attributes according to their affinity measure defined. A heuristic algorithm is applied for dividing the set of attributes into overlapping or non-overlapping fragments. Cornell and Yu developed an integer

programming formulation to solve the problem of VP[29, 30]. It works by splitting the relation into two fragments iteratively. The solution is found when there is no more profitable split can be resulted. However, this approach can only find a locally optimal partition.

Navathe and Ra [87] proposed the use of graph theory to solve the vertical partitioning problem. The major feature of this algorithm is that all fragments are generated by one iteration in a time of $O(n^2)$. In this approach, an associated attribute-affinity matrix is constructed based on an affinity matrix. The partition can be produced by identifying the distinguished cycle from the graph. A heuristic algorithm is used to find a cycle-atomic distinguished cycle in the affinity graph. However, the cycle-atomic completely distinguished cycle cannot be found even if it exists. Later, Lin and Zhang [79] proposed a new graphical approach to overcome the deficiencies found in the Navathe and Ra algorithm. In the new approach, it also works with the associated attribute-affinity graph. However, the fragments are identified by finding the 2-connected atomic distinguished sub-graphs instead of the cycle-atomic distinguished cycle.

Cheng et al[23] proposed to formulate the VP problem to the associated traveling salesman problem by the distance measures among the attributes and transactions. The genetic algorithm based algorithm is also introduced to solve the associated problem. However, the diagonal block structure of the transaction / attribute matrix cannot be guaranteed, which makes the partitioning scheme difficult to be determined as discussed in the previous chapter.

4.3 Clustering XML documents

In the relation database design, we can make use of the Entity-Relation Model [95, 102] for the data modeling. A conceptual schema is designed by using an entity-relationship diagram (ERD) and certain guidelines developed for the modeling. The schema is then mapped to a set of relations and would be materialized to be the database system. Normally, the database system does not suffer from any data redundancy and anomalies if appropriate data modeling is applied. However, the database model would contain some undesired features, especially when the database model is large and complex. As a result, the normalization to the entity-relationship model is introduced to avoid the anomalies by the decomposition of the entities if necessary.

For the XML documents, the schematic information may be made available when the document is built. Document Type Descriptors (DTD)[13] and XML schema [97] are the example formats of the schematic information. However, the schematic information is not necessary to be included in each XML document[89]. Also, for the same domain of information to be represented, the XML documents would have similar structure. However, with the structures being designed by different designers, the schematic information for the documents would be represented in different ways. Data integration is necessary for the heterogeneous collection of XML documents[75]. Xtract [47] is an automatic system for extracting the schematic information from a collection of XML documents. However, it could not distinguish the heterogeneous groups of XML documents within the collection. This results in the evaluation of structural similarity in XML documents [89] as well as the

clustering XML documents techniques [78]. By clustering the XML documents by structure, groups of XML documents with similar structures could be identified. The schematic information inferred would be better suit for the whole collection of the XML documents.

In fact, considering the normalization of ER-model and clustering techniques of XML documents by structure, both have many similar characteristics. For a relational data model, each entity may be regarded as a container with attributes to describe it. The entity relationship model is a model for defining the relationship among the entities (things) in the database system. For the XML document, the schema defines the structure, content and semantics of the XML documents. By clustering the XML documents by structure, groups of XML documents with similar structure would be formed. The general schemas for the set of groups of XML documents could be inferred by the automatic extraction system as discussed before. For the design of relational data model, the entity-relationship model would be available, but the model might be inappropriate. In the same sense, the schematic information, such as DTD or XML schema, would be available, but they are not the most general form for representing the collection of the XML documents. The introduction of clustering XML documents by structure could also be regarded as the normalization of schematic information for the XML documents, which is analogous to the normalization of schema (entity-relationship) in the relational database modeling. The normal form of the relational model could avoid the data redundancy and anomalies while the general schemas could allow a search to only relevant portions of the data, which results in higher efficiency.

For the physical design of a relational database system, once the statistics about the user access patterns are known, the performance of the database system may be improved by vertical partitioning. For the vertical partitioning problem, by using the information about the attributes involved for each access pattern (transaction), the attributes used together frequently are grouped together so that only the relevant portion of the information are retrieved for the transactions. This results in the reduction of the disk access in the database system and hence the efficiency could be improved. For the XML repositories, similar partitioning approach could be applied to improve the performance of the XML database system. By using the information about which XML documents are involved in each transaction, the XML documents which are frequently accessed together should be grouped to be a cluster in the database system. As a result, when the transaction is in process, only the relevant cluster would be in consideration and hence the performance of the query would be improved. Since the query expression for the XML documents is more sophisticated than the SQL query expression for the relational database, a search on only relevant data is in crucial for improving the efficiency of querying XML documents.

From the above, we could see that clustering XML documents is very important for the XML database design. In the following section, we would introduce the use of BTACO-based clustering for clustering XML documents by structure as well as by transaction pattern.

4.4 Proposed approach using BTACO-based clustering

In the previous chapter, we have introduced the Bi-Tour Ant Colony Optimization clustering method for the block clustering, in which the source data consists of a set of objects and a set of variables. The performance of the proposed method is also tested by the case of vertical partitioning problem. In this section, by means of the property of XML repositories and the XML documents themselves, we introduce the use of BTACO-based clustering for clustering XML documents by structure as well as by transaction pattern.

4.4.1 Clustering XML documents by structure

Lian et al [78] proposed to measure the similarity between pair of XML documents by using the following formula:

$$dist(C_1, C_2) = 1 - \frac{|sg(C_1) \cap sg(C_2)|}{\max\{|sg(C_1)|, |sg(C_2)|\}} \quad (4.1)$$

where $sg(C_i)$ denotes the semi-structure graph representing the corresponding XML document C_i. $|sg(C_i)|$ is the number of edges in $sg(C_i)$ and $|sg(C_1) \cap sg(C_2)|$ is the number of common edges between $sg(C_1)$ and $sg(C_2)$.

The distance defined is a metric based on common maximal sub-graphs [15, 41]. The more the number of common edges, the higher is the degree of similarity between the graphs. In the XML case, it indicates that there are more common element-subelement relationships between the pair of XML documents. As a result, there would be more path expressions which can be answered by both XML documents and hence it would be better to store both XML documents in the same

cluster for improving the performance of the XML repository.

The proposed metric also has several advantages over the edit-tree distance measure. It could prevent a semi-structure graph which is a sub-graph of another semi-structure graph from being "swallowed". For the edit-tree distance measure, if a certain graph is a sub-graph of another graph, the resulted distance would be smaller as less steps are required for the transformation is in such case. For considering the larger graph as the source tree, the transformation could be achieved by pruning larger portion of the irrelevant branches in each step and hence only a small number of steps are required. In the case of common-edge measure, since the denominator depends on the documents with larger number of edges, the "swallowed" effect would be reduced in the measure. Also, as finding out the common edges require only one scan of all the XML documents, the distance measure could be found in a more efficient way. Lian et al[78] also proposed using a ROCK-derived clustering method, named "S-GRACE" for the XML clustering task. ROCK[51] is a categorical clustering method which measures the similarity among the clusters relative to the number of common neighbors. The higher the number of neighbors, the higher is the similarity between the pair of clusters. S-GRACE[78] improved the ROCK by resulting in a set of more evenly distributed clusters.

Because of the advantages of using common-edge measure, we also apply the common-edge measure as the basis in the BTACO-based clustering XML documents by structure. Since BTACO-based clustering is a block clustering method, which requires the source data with a set of objects and a set of variables, we have to define the objects and variables before the algorithm can proceed. Here, the set of objects

would be the collection of the XML documents while the set of variables would be the collection of edges involved in the whole set of XML documents. Since the number of documents in the whole collection is very great, we have to perform pre-process work to reduce the number of semi-structure graphs to be considered. The number of semi-structure graph candidates could be largely reduced by considering the graphs with the same set of edges together during clustering[78]. As the metric considers the sources and destinations of the edges only, the cardinality of the element-subelement relationship is not in the consideration. The repetition of the relationship in the structure is also not in the consideration. As a result, the algorithm only clusters the semi-structure graphs with unique set of edges, which can perform the clustering in a more efficient way.

After the reduction of the candidates of the graphs for clustering, we have to define the data matrix for the block clustering. A semi-structure graphs / edges matrix would be the source data for the clustering. In the data matrix, the sets of semi-structure graphs would be the elements in the column while the sets of edges would be the elements in the row. In our approach, we extend the work of Lian et al by introducing the weighting for each edge in the data. The weighting for each edge is equal to the number of documents involved in the collection.

$$Weighting_{Edge(i)} = \{\#\text{ of documents involved}\} \quad (4.2)$$

By the introduction of weighting for each edge, the outlier edges would have less effect on the clustering process. On the other hand, the edges with higher degree of importance can be used for clustering the collection more accurately. In our proposed approach, we would introduce the same distance measure, City-block

distance measure as discussed in the previous chapter for the block clustering.

In the experiment part, we will use both the real data sets and synthetic data sets with the cluster information given. We will compare the accuracy of the proposed approach by calculating the Hubert statistic index (explained in the experimental part) for each data set. However, since the original BATCO-based clustering algorithm is not compatible for categorical clustering, we have to modify the algorithm slightly so as to support the categorical clustering.

As discussed in the previous chapter, the BTACO-based clustering involved two parts, the permutation of row and column order and the identification of the sub-blocks. To facilitate the categorical clustering, only the second part would be modified. There are two approaches for the identification of the sub-blocks. As the heuristic partitioning approach is more efficient than the dynamic programming approach, we would discuss about the heuristic partitioning approach only.

Considering the heuristic partitioning approach, the algorithm would bi-sect the row and column simultaneously to form four matrices in each step iteratively, with the left-top and right-bottom matrices would be further divided until there is no profitable division found. If we merely restrict the level of the iterative algorithm, the number of category can only be restricted in the order of 2, say 2, 4 and 8, etc. As a result, we have to modify the algorithm in a different way. In the modified algorithm, all the favorable cutting positions are recorded with the associated value of reduction in the weighted sum of error (WSE). The cutting positions would be sorted by the ascendant order of the amount of reduction. For finding out k

categories, (k-1) cutting positions would be chosen. It would be a great idea to consider the top-(k-1) cutting positions for the clustering result. However, there would have a case in which the cutting position is defined in a sub-matrix which is not defined at all. Since the nature of the heuristic partitioning approach is in iterative, if we merely consider the top-(k-1) cutting positions, since the reduction in the deeper iteration might be in greater value in the same hierarchical division of the sub-matrix, it is not reasonable to consider the cutting positions merely. As a result, we have added a constraint to the algorithm in which the cutting position is feasible only when its corresponding un-partitioned sub-matrix is defined by the previous division. This cna be achieved by assigning an id number to each cutting position. Since the nature of the heuristic partitioning approach is a binary division algorithm, we can assign the id number to each cutting position by modifying the algorithm as follow:

```
bi-cut (id, start_col, end_col, start_row, end_row) {
    ……
    ……
    bi-cut (2 x id, ……)
    bi-cut (2 x id + 1. ……)
}
```

We have added the parameter *id* as an additional input parameter in the bi-cut algorithm. The *id* is then assigned to each cutting position iteratively as shown in the pseudo code. By this assignment, the candidate cutting position with id number *id* is feasible if the cutting position(s) with id number equals to *parent_id* exist, in which the id *parent_id* should satisfy the following two properties:

$$Parent\_id = \left\lfloor id \times \left(\frac{1}{2}\right)^{i} \right\rfloor, \forall i \geq 0 \qquad (1)$$

$$Parent\_id \geq 0 \qquad (2)$$

For finding out the k categories in the clustering algorithm, we first find out all the candidates of the cutting position by the BTACO-based clustering algorithm. Then we sort the candidates with the descendant order of the reduction in WSE, i.e. the one with the largest reduction should be in the top rank. For getting k clusters, we would iterate k-1 times to get the most appropriate cutting position by maximizing the reduction of the WSE. This could be done by choosing the feasible candidates with the largest reduction. The feasible cutting position is the cutting position in which its corresponding sub-matrix is formed. In the implementation of property (1), we need to check whether the cutting position with id = id/2 exists only, as if this cutting position does not exist, its all ascendants, which is the super-matrix of the corresponding sub-matrix, should also not exist. For property (2), it is used to guarantee that the first cutting position from the heuristic partitioning approach, i.e. the cutting position with id =1 should be in the partitioning solution. The pseudo code for the choosing mechanism is shown in Figure 4-1.

```
initialize sort_list; // cutting position sorted in the descendant of reduction in WSE
for i = 1 to k -1 {
      current_rank = 1;
      while (cutting position not chosen) {
            candidate <- sort_list[current_rank];
            check whether candidate satisfy property (1) and (2)
            if (satisfied) {
                set of cutting position <- candidate;
                remove candidate from sort_list;
            }
            increment current_rank;
      }
}
```

Figure 4-1 Pseudo code for the mechanism of choosing (k-1) cutting position

From the pseudo code, the core part iterates (k-1) so that (k-1) cutting position will be chosen. In the core part, there is a while-loop to scan the candidates from the one with the largest reduction in WSE. If the candidate is feasible, it will become one of the cutting positions in the clustering result and will be removed from the candidate

list (sort_list). The while-loop part will be run again with the while-loop scanning from the candidate with top rank again until k-clusters have been formed.

The experimental study of the proposed approach will be seen in the next section. In the experimental study, we will see that the BTACO-based clustering could result in a significant better result compared with the clustering based on the edit-tree distance measure. The result is also better than the clustering based on the common-edge metric using S-GRACE clustering method, the performance is better especially when there is a high degree of outliers in the collection.

4.4.2 Clustering XML documents by user transaction patterns

In the last part, we have shown the proposed approach for clustering XML documents by structure. In this part, we will show the proposed approach for clustering XML documents by user transaction patterns. For the proposed approach, we will also use the BTACO-based clustering method.

In the physical design of the relational database model, the vertical partitioning is applied when the statistics about the user query pattern is known. The attributes which are used together frequently would be grouped together so that the disk access could be minimized by accessing them together. Due to the significant improvement in the application of vertical partitioning in relational database system, we propose the similar application in the physical design of the XML database model.

In our proposed partitioning approach, the collection of XML documents would be divided into certain number of clusters so that the portion of XML documents which are accessed together would be in the same group. As a result, the query would search only the irrelevant portion of the data in the database, which results in great improvement in the efficiency. To facilitate the partitioning, we introduce the XML documents / query patterns matrix for the database design profile. In the matrix, the column set is the collection of XML documents while the row set is the collection of query patterns with the access frequency given by the statistics. Each element in the matrix A_{ij} is defined as follow:

$$A_{ij} = \begin{cases} 1, & \text{if query i have access in XML doc. j} \\ 0, & \text{otherwise} \end{cases} \quad (4.3)$$

In this way, we will know the portion of XML documents to be accessed for each query pattern. For each query, it is weighted by the access frequency. Due to the weighting, we can make a trade-off in which the portion of XML document collection which is accessed more frequently will likely be grouped together. With the matrix defined, we will use the block clustering method to determine the number of clusters as well as which XML documents should be in the specific cluster.

In the partitioned environment, we have to determine the partitions that contain the result in the given query. This job can be done by the query manager. The query manager stores the required information to determine which set of partition have to be searched in order to give out the result for the given query. Lian et al [78] have proposed using a set of bit arrays to store such information. The set of bit arrays is in fact a representation of the summary for the set of XML documents in the specific partition. As aforementioned, the XML documents can be treated as semi-structure

graphs, which are then represented by sets of edges identified in the graph structures. The query manager actually stores the union of all the set of edges for the XML documents, which is in the form of set of bit arrays. For each query, the expressions involved are parsed to be transformed into a set of bit arrays corresponding to the edges involved. Each query bit array is then compared with the bit arrays stored in the query manager by a 'And' Boolean binary operator. The partition should be involved in the query if there is no change in the bit array after performing in the 'And' operation. Otherwise, the partition should not be involved.

In fact, the above approach is very efficient in dealing with the simple path expression, i.e. the expression involves "/node/node/...../node". However, the approach cannot be applied to the case in which the expression involves more complicated axes such as wildcards "*" and relative path "//". Lian et al [78] have proposed a solution in such problems by providing a method to transverse the whole root semi-structure graph, which is the graph representing the entire set of documents in the partition. For instance, consider the path involves "A//B", all the paths from A to B in the root semi-structure graph are identified by traversing from the starting node A to node B.

In this paper, we proposed applying the XML index structure approach for the query manager. The XML index structure is an index for locating where required data resides in an efficient way. Since graphically, the XML document is in the ordered labeled tree structure, the index should be able to handle the structural information and the content as well as the relationship among the structure and the content. There have been many literatures proposed[19, 56, 67, 68, 77, 85, 94, 101, 106]. In

general, there are two approaches in the indexing techniques, they are the region-code based approach and the path based approach. For the region-code based approach, each node in the XML document is assigned with a triplet index, usually in the form of <start, end, level>. In this way, considering two nodes, say A and B, the ancestor / descendant relationship can be identified by comparing whether "A.start < B.start < A.end" is satisfied. If the statement is satisfied, then B is the descendant of A. The parent / child relationship can also be identified by checking with the previous statement and "A.level = B.level -1". If both statements are satisfied, then B is the child of A. The absolute region code (ARC)[106], the relative region code (RRC)[68] and <order, size> numbering scheme [77] are example approaches in the region-code based indexing. For the path based approach, a set of indices is assigned to each possible path for the XML document. Since the number of possible paths is very large for the normal sized XML document, the path-base approach indexing requires the technique in reducing the number of indices by eliminating the duplicated information or only considering the most-frequent used information. In general, the overall set of indices form a structural summary of the XML document which is represented as a data graph. There are several methods used this approach, such as DataGuides [48], 1-Index [85], A(k)-Index [67], D(k)-Index [19] and M(k) / M*(k) index[56], etc.

In our partitioning approach, we use the indexing technique for testing whether the XML documents in certain partition matches the query. The proposed query manager works as follow:

- (Summary graph structure) For each partition, a summary graph structure is formed by the union of all the graph structures which are used to represent the

structure of all the XML documents. The summary graph structure is then indexed for efficiency checking. Since the summary graph structure is used to test whether the query matches the XML documents inside certain partition, the exact location for the result of the query is not necessary, the path based coding is not useful here.

- (Region code based indexing) The region code based indexing is introduced. All the nodes in the summary graph structure are indexed with the triplet index and are then stored in a hash-table for efficient look-up.
- (Query parsing) The XPath[25] query will be parsed in the query manager. For the step in the XPath which involves ancestor / descendant or parent / child relationship, the corresponding triplet indices of the involved nodes are fetched from the index table in each partition. If any of the relationships could not be found in certain partition, the partition will not be included in the revised query. The partition will only be involved if all the relationships matches with the one expressed in the XPath query.
- (Query rewriting) After determining which sets of partition are required, we can re-write the query to suit in the partitioned database system. This can be done by simply adding the partition name as the prefix of the query. For instance, "/A//B/C" will be re-written as "partition_name/A//B/C". If more than one of the partitions are chosen. The query will be rewritten as the union of the revised query with different partitions in each revised query.

4.4.3 Implementation of Query Manager for our experimental study

In this part, we will show the details of implementing the proposed query manager for the experimental study. As aforementioned, there are four stages of work in the query manager, in which the first two stages are the stages for initializing the index table for query parsing in the query manager. In this part, we will first discuss about the initialization of the index table from the collection of XML document and we will introduce a possible way of parsing the query expression to determine the partitions involved and re-writing the query expression for the database to query the result.

Initialization of index table

To determine which partitions to be involved for the query process, an index table is necessary for each partition. The index table can be used to validate whether the structural query meet the structure of the XML documents in the specific partition and hence the specific partition should be searched through. To build the index table, first we have to determine which documents should be grouped together. This could be done by the clustering method as discussed before. Second, a summary graph structure can be formed by traversing all the XML documents in the partition. If certain node of the structure is not included in the summary graph structure, the node will be added to corresponding of the summary structure. In initial, since the root elements in the XML documents may not be the same, we have introduced a "virtual" root in the summary structure and so all the unique XML structures should

be the child of the virtual root. We applied the coding scheme proposed in [106] for our query manager. For each partition, the coding scheme is applied to the summary structure and an individual index table (as inverted list) is formed.

Parsing and Re-writing query expression

In the query manager, the index tables built are loaded into memory for efficient look-up. Also, the index tables are stored as the hash tables with the tag name as the key and the triplet index <start, end, level> as the value. In our experimental study, the query manager parses XQuery [91] expression with the following structure:

```
for (variable) in collection('/db/')[XPath expression]
where [XPath expression] op [XPath expression] op .....
return (variable)[XPath expression]
```

In the proposed parsing method, we first locate the variables (e.g. $x) and their associated domains, i.e. collection('/db/')[XPath expression] as shown in the above. The nested for-clause could also be supported. Then we determine the partitions to be involved by parsing the where-clause statement in the XQuery expression. In the parsing method, the sets of XPath expression are first extracted from the where-clause statement. The XPath expression are then further parsed for determining the set of tag names as well as the set of structural relationships to be considered. The variable associated with certain XPath expression is also determined. Afterwards, the query manager validates the set of tag names as well as the set of structural relationships (either absolute location path or relative location path) for each partition by looking up the hash tables. In our query manager, only the ancestor-descendant and parent-child relationship are considered as they are the most frequent used structural relationship for XML query. The partitions will not be included if any validation test fails. For the set of XPath expressions, since there

might be more than one XPath expression associated with the same variables, we have used the bitwise expression to store the set of partitions involved for the variables. The finalized set of partitions are determined by simply performing the "AND" operation of all the bitwise. Finally, after we have determined the set of partitions for each variable. The query manager would re-write the for-clause statement such as each variable would search the feasible set of partitions. The pseudo codes for the above processes are shown as in Appendix III. The flow of our proposed XML database system for partitioned XML collection is shown in Figure 4-2.

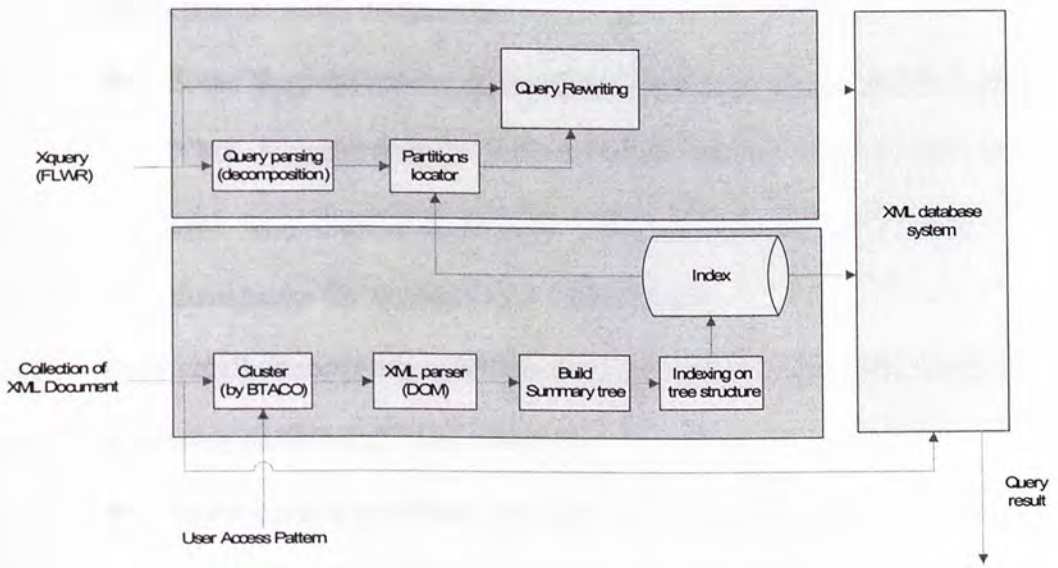


Figure 4-2 Framework of the database system for the partitioned XML collection

There are advantages with our proposed partitioning approach:

1. Co-existence with any storage model and indexing structure
    - Co-exist with different storage model. Since the partitioning model is to allow the search to access the more relevant portion of the data so as to improve the performance, in which it could be done by altering

the query to access certain set of partitions. It is not necessary to modify the storage model so as to apply our proposed partitioning approach.

- Co-exist with different indexing techniques. There have been many literatures about the indexing techniques for the XML documents[20, 50, 68, 77, 99, 101, 106]. Similar to the case of the storage models, the indexing techniques could be applied to the partitions of the XML documents.

2. No alternation of the structure and content for storing partitioned collection of XML documents
   - Since the partitioning approach is based on storing the collection of XML documents in the different tables instead of storing them in one table only, there is no need to change the structure of the XML documents for facilitating the partitioning.
3. Only small changes required in the original XPath query expression for accessing partitioned XML collection
   - For the query rewriting, we only have to add the collection name as the prefix of the XPath expression. For the query which involves several partitions, the query result would be the union of all the queries from XPath expressions with the prefix as the name of the collection (partition) involved.

In the experimental study, we will show the performance of the partitioning approach by comparing the performances on different types of queries with both the un-partitioned and partitioned case.

4.5 Experimental Study

In this section, we would study both the performance of the BTACO-based clustering of XML documents for by structure and by user transaction patterns. We will discuss about the experimental study on the clustering by structure first. And then we will discuss about the study on the clustering by the user transaction patterns.

4.5.1 Experimental Study on the clustering by structure

In this part, we will first discuss about the data sets we have used to evaluate the performance of various clustering method for clustering XML documents by structure. Then, we will compare the results by BTACO-based clustering method with the clustering methods using edit-tree distance and the common-edge distance respectively.

4.5.1.1 Data Sets

In the experiment study, we test the performance of the clustering methods by both real data set and synthetic data set. For the real data set, the SIGMOD 2002 xml data collection is used. For the synthetic data set, we use a XML generator to generate a collection of XML documents based on the pre-defined parameters.

Real data set

For the real data set, we have used the online version of the ACM SIGMOD Records 1999[3]. In the data set, we have to test the accuracy in clustering the XML documents into four different sets. The information about the dataset is shown in Table 4-1 Information about the ACM SIGMOD record in xml version.

Synthetic data set

For the generation of synthetic data, we have developed an XML documents generator to generate a collection of XML documents in which fixed number of groups is specified. The XML GENERATOR[33] and ToXgene[8] are tools for generating the XML documents based on the user defined schematic information. However, as stated by Lian et al[78], the tools are lack of the ability in controlling the cluster distribution and similarity automatically. As a result, an XML generator had to be built which can handle the cluster distribution and similarity. Our generator differs from the Lian et al version by several differences in building the schematic information for the generation of XML documents.

- Instead of using the NITF (News Industry Text Format) DTD [2] as a seed DTD for generating the schemas, we have generated the schemas randomly, based on a set of fixed number of keywords. In the former generator, it uses the seed DTD to extract many sub-DTDs in which they have a certain degree of overlapping. In our generator, the schemas are randomly generated based on a fixed set of keywords for the tags.

- To generate the overlap schemas, the former generator generates by extracting the overlap sub-DTDs from the seed DTD. In our generator, we generate the

overlap schemas by generating the structure based on a certain set of keywords for the tags formation in the schemas.

- Instead of evaluating the degree of overlapping of XML documents by the proportion of common-edge, we have used the proportion of common tags as the evaluation. If both the measurement and the generation of synthetic data are based on the same criterion, it would be susceptible in whether the data set is in favor of clustering by using common-edge distance measure.

Due to the above issues, we have built an XML documents generator which is a three-step process system as follow:

1. Given the expected number of documents in each cluster, the expected depth as well as the expected breadth of the tree structure as the input parameters, the schemas are formed as the tree structure with the cardinality of the tag element defined. The depth of the schema is defined as the depth of the tree structure in the schema. The breadth of the schema is defined as the number of tags to be under a certain node. The depth and breadth are based on the Poisson distribution with the expectation values given. In general, the depth of the schema is not too high and we set the value to be 4 in default. Also, the number of child node for each node would not be too high and we set the value to be 6 as default value. For each cluster, the number of documents is determined by the Gaussian distribution with mean and variance defined as:

$\mu = \#doc.$ in each cluster

$\sigma = \frac{1}{3} \times \#doc.$ in each cluster

2. Given the degree of overlap as an input parameter, we get two schemas (say, A and B) from the set of generated schemas. We build a new schema in which the tag elements are decided by a uniform random process *P*. If P < deg. of overlap, the tag elements would be the list of tag elements from schema B. Otherwise, the tag elements would be the list of tag elements from schema A. After the formation of the new schema, the exist schema A will be replaced by the new schema as one of the member in the set of generated schemas. The above process will be repeated for twice the number of clusters times to make a chance that every schemas will have certain degree of overlap with the others.

3. After the generation of the schemas, we will build the XML based on the schemas generated. In order to add some outliers in the collection, we will perform "mutation" process when building the XML documents. Given the degree of outliers (O) as an input parameter, there will be a chance of O in which the name of the specific tag will be altered. Also, with the same chance, a new tag will be added to the XML documents.

We have generated six different synthetic sets of XML documents for the experiment. The input parameters for each synthetic collection are shown in Table 4-2. By generating the collection of XML documents by the input parameters shown

in Table 4-2, we have built 18 different datasets with the information shown in Table 4-3. The datasets differ from the degree of outliers, the degree of overlapping and the number of expected clusters. Also, we have generated two datasets for each configuration. There are totally 9 different configurations in our experiment. The table shows the actual total number for the whole dataset, the number of unique bitwise structure and the number of unique edges identified in the XML structures (explained in previous section) for each collection of XML documents.

Number of clusters:	4
Total No. of doc:	986
No. of unique bitwise structure:	5
No. of edges identified	54

Schemas	No. of xml
SigmodRecord	1
ProceedingsPage.dtd	17
IndexTermsPage.dtd	917
OrdinaryIssuePage.dtd	51

Table 4-1 Information about the ACM SIGMOD record in xml version

	Set 1(*a/b*)	**Set 2(*a/b*)**	**Set 3(*a/b*)**	**Set 4(*a/b*)**	**Set 5(*a/b*)**
Expected # of doc / cluster	100	100	100	100	400
No. of clusters	5	5	5	5	5
Degree of overlapping	0.5	0.5	0.9	0.9	0.5
Expected depth	4	4	4	4	4
Expected breadth	6	6	6	6	6
Degree of having outliers	0.001	0.01	0.001	0.01	0.01

	Set 6(*a/b*)	**Set 7(*a/b*)**	**Set 8(*a/b*)**	**Set 9(*a/b*)**
Expected # of doc / cluster	100	100	100	100
No. of clusters	10	10	10	10
Degree of overlapping	0.5	0.5	0.9	0.9
Expected depth	4	4	4	4
Expected breadth	6	6	6	6
Degree of having outliers	0.001	0.01	0.001	0.01

Table 4-2 Table shows the input parameter setting for each synthetic collection

Set	**1a**	**1b**	**2a**	**2b**	**3a**	**3b**	**4a**	**4b**	**5a**	**5b**
Total No. of Documents	446	398	546	520	523	461	447	492	2126	2331
No. of unique bitwise structure	37	37	46	67	134	147	98	179	36	403
No. of edges identified	89	87	106	268	429	211	168	616	106	782

Set	**6a**	**6b**	**7a**	**7b**	**8a**	**8b**	**9a**	**9b**
Total No. of Documents	516	465	432	450	560	463	538	386
No. of unique bitwise structure	59	51	77	66	159	105	158	192
No. of edges identified	152	121	282	316	317	263	581	593

Table 4-3 Table shows the number of documents, the number of unique bitwise structure as well as the number of edges identified for each cluster in the synthetic collection

4.5.1.2 Computing Environment

All the experiments were done on a PC with processor Intel Pentium 4 running at 2.8 GHz. All three clustering methods were implemented by using Java language, except the clustering process in the hierarchical clustering with edit-tree distance measure, which was done by using Matlab. In the study of timing results, we have provided the pre-processing time and the clustering time.

4.5.1.3 Evaluation

In the experimental study, we will compare the performance of three different clustering methods; they are the proposed BTACO-based clustering, the hierarchical clustering method with edit-tree distance measure and the SGRACE[78] clustering method with common-edge distance measure. For the hierarchical clustering methods, we use the Unweighted Pair-Group Averaging method (UPGMA) for the hierarchical clustering. The same clustering approach had been also applied in [89] for evaluating the edit-tree distance measure. For UPGMA, the distance between cluster C_i and C_j is computed as:

$$Dist(C_i, C_j) = \frac{\sum_{k=1}^{|C_i|}\sum_{l=1}^{|C_j|}\delta(doc_k^{C_i}, doc_l^{C_j})}{|C_i||C_j|} \quad (4.4)$$

where $|C_x|$ is the number of XML documents contained in cluster C_x and doc_k^{Cx} is the k^{th} XML documents in the cluster C_x.

Parameters	Values
Iteration (I)	50
Alpha(α), Rho(ρ)	0.9
Beta(b)	3.0
Detla(d)	0.3
Q_0	0.9

Table 4-4 Parameter setting for the modified BATCO-based clustering.

To compare the performance, we will use the modified BTACO-clustering method to generate the set of clusters in which the number is equal to the clusters given in the data set. We will use the parameters shown as in Table 4-4. In the same way, by means of the UPGMA distance measure among the clusters, we will identify the clusters in the hierarchical clustering result in which its number is equal to the number defined in the data set.

To evaluate the performance, we will use the Hubert Γ statistics[66] for the evaluation. The Hubert Γ statistics measures how well of the result in the clustering method correlated with the pre-specified clustering structure. In the statistics, the index ranges from 1 to -1 in which the value near 1 indicates the high correlation between the two clustering structure. On the other hand, the value near -1 indicates the low correlation which means that the clustering structure resulted by the clustering method is poor. For the Hubert Statistics, the index is defined as:

$$\Gamma = \left[\frac{1}{M} \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} (X(i,j) - \mu_x)(Y(i,j) - \mu_Y) \right] \Big/ \sigma_x \sigma_Y \qquad (4.5)$$

where

N is the total number of (unique) XML documents

M = n(n-1)/2

$$\mu_X = (1/M) \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} X(i,j) \ , \ \sigma_x^2 = \mu_X - \mu_X^2$$

$$\mu_Y = (1/M) \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} Y(i,j) \ , \ \sigma_Y^2 = \mu_Y - \mu_Y^2$$

$$X(i,j) = \begin{cases} 1, & \text{if i and j are in the same cluster} \\ 0, & \text{otherwise} \end{cases}$$

$$Y(i,j) = \begin{cases} 1, & \text{if i and j are in the same cluster} \\ 0, & \text{otherwise} \end{cases}$$

Refer to the formula, the X(i,j) and Y(i,j) are known as the proximity matrices which define whether the objects i and j are in the same cluster. The variables μ_x, μ_y, σ_X^2, σ_Y^2 are the mean and variance of the values lies on the proximity matrices X(i,j) and Y(i,j) respectively, which aims at normalizing the Hubert index and so it would only ranges from -1 to 1. For comparing the clustering methods, the higher the value of the index, the better is the clustering method. The result of the performance study is shown in Table 4-5.

To compare the effectiveness of the algorithms, we compare the computation time for clustering the XML documents by three different methods. Refer to Table 4-6 to Table 4-8, it shows the time required for pre-processing and clustering in hierarchical clustering method with edit-tree distance measure, S-GRACE method and BTACO-based clustering respectively. The pre-processing time is the sum of the time for parsing the collection of XML documents to the desired structure and the time for calculating the distance measures among pairs of XML documents in the

collection. The clustering time is the time for finding out the clusters based on the methods and the distance measure provided. For clustering method with edit-tree distance measure, the pre-processing time is the time for finding out the edit-tree distance among pairs of XML documents using dynamic programming approach. For the S-GRACE clustering, the pre-processing time is sum of the time for parsing the XML documents into sets of bit-arrays and pruning out the replicated bit-array structure if necessary as well as building the similarity matrix for clustering. Finally, for the BTACO-based clustering, the pre-processing time is the sum of the time for parsing the XML documents as in S-GRACE as well as the time for building the XML-documents / Edges matrix for clustering. The total computation time for each clustering method is summarized in Table 4-9 for comparison.

SET	SIG-MOD	1a	1b	2a	2b	3a	3b	4a	4b	5a	5b
Edit-Tree	0.48	0.78	0.26	0.70	0.02	0.052	0.08	-0.014	0.04	0.20	0.35
S-GRACE	1.00	1.00	1.00	0.68	0.81	0.67	1.00	0.67	1.00	0.76	0.39
BTACO	1.00	0.99	0.99	0.94	1.00	0.82	1.00	0.76	1.00	0.98	0.99

SET	6a	6b	7a	7b	8a	8b	9a	9b
Edit-Tree	0.29	0.35	0.17	0.23	0.10	0.16	0.24	0.09
S-GRACE	0.89	0.75	0.98	0.72	0.78	0.89	0.74	0.74
BTACO	0.96	0.78	0.98	0.79	0.81	0.99	0.83	0.79

Table 4-5 Comparison of Hubert indices for three different methods

SET	SIGMOD	1a	1b	2a	2b	3a	3b	4a	4b	5a	5b
Pre-processing	21079	1488	263	1873	390	1063	3322	3583	2357	5950	23011
Clustering	3303	91	72	219	215	219	128	114	168	76357	110410
Total Time:	24382	1579	335	2092	605	1282	3450	3697	2525	82307	133421

SET	6a	6b	7a	7b	8a	8b	9a	9b
Pre-processing	562	247	255	212	2040	768	1365	1933
Clustering	210	136	101	119	302	133	253	64
Total Time:	772	383	356	331	2342	901	1618	1997

Table 4-6 Computation time for each process in clustering using edit-tree distance (in sec)

SET	SIGMOD	1a	1b	2a	2b	3a	3b	4a	4b	5a	5b
Pre-processing	16	4	2	6	6	4	9	8	18	13	100
Clustering	1	1	1	1	3	72	87	24	180	1	9029
Total Time:	17	5	3	7	9	76	96	32	198	14	9129

SET	6a	6b	7a	7b	8a	8b	9a	9b
Pre-processing	4	3	6	5	13	6	18	16
Clustering	2	1	5	2	88	13	64	197
Total Time:	6	4	11	7	101	19	82	213

Table 4-7 Computation time for each process in using S-GRACE method (in sec)

	SIGMOD	1a	1b	2a	2b	3a	3b	4a	4b	5a	5b
Pre-processing	16	4	1	6	5	4	5	7	12	12	69
Clustering	10	57	14	93	332	1837	2758	99	1461	92	1841
Total Time:	26	61	15	99	337	1841	2763	106	1473	104	1910

	6a	6b	6a	6b	6a	6b	6a	6b
Pre-processing	4	2	5	5	9	4	13	10
Clustering	73	38	398	517	750	362	1486	1500
Total Time:	77	40	403	522	759	366	1488	1510

Table 4-8 Computation time for each process using BTACO clustering method (in sec)

	SIGMOD	1a	1b	2a	2b	3a	3b	4a	4b	5a	5b
Edit-Tree	24382	1579	335	2092	605	1282	3450	3697	2525	82307	133421
S-GRACE	17	5	3	7	9	76	96	32	198	14	9129
BTACO	26	61	15	99	337	1841	2763	106	1473	104	1910

	6a	6b	6a	6b	6a	6b	6a	6b
Edit-Tree	772	383	356	331	2342	901	1618	1997
S-GRACE	6	4	11	7	101	19	82	213
BTACO	77	40	*403*	*522*	759	366	1488	1510

Table 4-9 Comparison of total computation time of three different methods (in sec)

Evaluation of accuracy among clustering methods

From the results in Table 4-5, we can see that BTACO-based clustering out-performs the other clustering methods in general. It could result the set of clusters in higher accuracy compared with the other two methods. Among the other two methods, we can see that hierarchical clustering with edit-tree (ET) distance measure performs the worst among the three methods and S-GRACE performs the second best.

Considering the real dataset SIGMOD records in xml version, we can see that the S-GRACE and BTACO clustering methods can identify the set of clusters without any error while the hierarchical clustering method with edit-tree distance measure results in a relatively low accuracy. This can reflect that hierarchical clustering with edit-tree distance measure is not a good method for clustering XML documents by structure. The clustering method using common-edge measurement performs better in the real dataset studied.

Considering the synthetic datasets, we compare the results by using different datasets which differs from the degree of overlapping and the degree of outliers. As anticipated, the high degree of overlapping and outliers would reduce the performance of the clustering methods. Despite this, the BTACO-based clustering and S-GRACE clustering method could still get a relatively high value of Hubert index. The clustering with edit-tree distance results in poor performance when the dataset having high degree of noisy data. Again, this reflects that the edit-tree distance is not a good measurement for the similarity among the schemas of the XML documents. The measurement by means of common-edge has a higher accuracy in identifying the clusters as shown in the results of BTACO and S-GRACE clustering methods. Among the two methods, BTACO works better than S-GRACE clustering method. As aforementioned in the previous chapter, considering the problem as a two-dimensional matrix could minimize the effect of noisy data in the similarity measure. Also, the weighting of the edges in the collection of XML documents could be regarded as the degree of importance of the edges identified. By means of considering the importance of the edges, the BTACO clustering method can result the set of clusters with higher accuracy. Because of the two reasons, the BTACO could results in good performance.

Evaluation of effectiveness among the clustering methods

Refer to Table 4-9, when compared with the computation time among the three methods, S-GRACE can find out the clusters effectively among the three methods.

As anticipated, the clustering with ET distance measure is much slower than the others, especially when the dataset is large, say 2,000 documents as in the synthetic dataset 5. The BTACO-based clustering can find out the clusters with reasonable time in general, except for some synthetic set such as set 3a, 6a and 6b, in which the reason will be discussed later.

Since for the hierarchical clustering with edit-tree distance, it has to compare all the pairs of XML documents in the collection, it spent much more time than the other two methods, in which only the unique structures based on edge have to be considered. Refer to Table 4-1, there are totally 986 documents in the XML collection, there would be 485,605 (986x985/2) pairs of XML documents to be evaluated for the edit-tree distance measurement. However, for the other two methods, there are only 5 unique structures identified and hence only 10 (5x4/2) pairs have to be evaluated. Refer to Table 4-3, the same situation occurs in which the pair of structures to be evaluated in the edit-tree distance is much larger than that in the other two clustering methods. The measurement by means of common-edge provides an effectiveness way of evaluating the XML documents by structure. Also, as discussed before, the time complexity of evaluating the edit-tree distance requires $O(n^2)$ for each pair of XML documents. On the other hand, for the S-GRACE method, the time complexity is O(1), as it requires the time for the AND binary operation for each pair of XML documents only. For BTACO-based clustering, the distance measure is included in the clustering method and so no comparison can be concluded.

For the comparison of the clustering time, S-GRACE results the set of clusters very

effectively. BTACO clustering works well in general except for the dataset #3a, #6a and #6b. The long computation times are due to the large number of identified unique structures and edges in the collection. For the BTACO method, it will generate a number of agents (ants) to search the result, in which the number would be equal to the sum of the number of identified unique structures and edges in the collection. The number of agents in fact equals to the sum of the row and column of the XML documents / edges matrix. As a result, for a large matrix, it will require much more time to get the result. Despite, in general, BTACO works faster than the hierarchical clustering with edit-tree distance, in which the computation time depends on the number of documents. For the collection with over 2000 documents (for example, considering data set #5a), it requires 76,357 (over 21 hours) to finish the clustering. The time is in fact unacceptable when compared with S-GRACE and BTACO methods, in which they require 1 and 92 seconds respectively. Although BTACO cannot be as effective as S-GRACE, there is a trade-off in getting the result accurately and effectively. BTACO can get the set of clusters more accurately even though the computational time is longer. However, even if the worst case as in dataset 3a for instance, it requires only nearly half an hour to finish the clustering. In fact, the computational time is acceptable as clustering process is not a contiguous process in the database system. It is run whenever the database system has to identify the schemas of the XML documents. Since it is run in off-line, the computational time which requires half an hour is acceptable.

4.5.2 Experimental Study on the clustering by user access patterns

In this experiment, we will study the performance of clustering method in partitioning the collection of XML documents based on the frequent transaction patterns; we will study the improvement in querying the XML documents. To compare the performance, we have used the un-partitioned collection as the control experiments. We will study two different partitioned cases to see how our clustering method partitions the collection so as to facilitate the improvement of querying the XML documents.

4.5.2.1 Computing Environment

To study the performance of the clustering method, we will study the improvement in querying the XML documents in the collection. All the tests were done on a PC with 1g RAM and Intel Pentium 4 processor, running at 2.8gHz. We will use the eXist[1] which is a java-based open source native XML database as the database system for managing the XML collections. The query manager and query re-writing process are implemented by us in Java.

4.5.2.2 Dataset

We have used the XML version of ACM SIGMOD record[3] as our dataset for studying the performance. The information about the dataset can be in Table 4-1. Since there are 986 XML documents, as discussed before, we will

consider the set of unique structures derived from the collection XML documents, resulting in 5 different bitwise structures only.

To study the performance, two difference sets of transaction patterns are used. The two different sets of transaction patterns have the same set of XQuery expressions and differ from the access frequencies of the expressions. In Figure 4-3 and Figure 4-4, the two matrices are shown as the access profile, in which they indicate which set of XML collections accessed by each query expressions given. In the figure, the left access profile is assigned as profile A and the right access profile is assigned as profile B.

Trans.	XML structure:					Acc. Freq
	a	b	c	d	e	
1				X		10
2				X	X	30
3	X	X				10
4			X			10

Figure 4-3 Matrix for access profile A

Trans.	XML structure:					Acc. Freq
	a	b	c	d	e	
1				X		30
2				X	X	10
3	X	X				10
4			X			10

Figure 4-4 Matrix for access profile B

Considering the transaction (row) of the matrix, we have used the id numbers from 1 to 4 to represent the pattern of transactions. Their corresponding XQuery expressions for the transactions are shown in Table 4-10.

ID	XQuery expression
1	for $x in collection('/db/Experiment/all') where $x//categoryAndSubjectDescriptorsTuple/content &= "Data Applications" return $x//title
2	for $x in collection('/db/Experiment/all') where $x//abstract &= "shared memory database systems" return $x//title
3	for $x in collection('/db/Experiment/all') where $x//articlesTuple/initPage < 45 return $x//title
4	for $x in collection('/db/Experiment/all') where $x//SigmodRecord/issue/volume = 11 return $x//issue/articles

Table 4-10 Representation of the XQuery Expression in the access profile

Trans.	XML structure:					Acc. Freq
	e	d	a	b	c	
2	X	X				30
1		X				10
3			X	X		5
4					X	5

Figure 4-5 clustering result for matrix in Figure 4-3

Trans.	XML structure:					Acc. Freq
	e	d	a	b	C	
2	X	X				10
1		X				30
3			X	X		5
4					X	5

Figure 4-6 clustering result for matrix in Figure 4-4

Figure 4-5 and Figure 4-6. Results of clustering the collection by BTACO-method

4.5.2.3 Use of BTACO clustering method

To facilitate the partitioned collection of XML documents, we have to cluster the collection in the database system. In our experiment, we have used the BTACO-based block clustering method to cluster the collections based on two different access profiles as shown in Figure 4-3and Figure 4-4. We have used the

same set of parameters for clustering by structure (refer to
Table 4-4) and the clustering results are shown in Figure 4-5 and Figure 4-6. From the results, we can see that different access profile results in different partitioning structures. For the profile A, there would be 3 partitions resulted with the total weighted sum of error (WSE) equals to 10. For the profile B, there would be 4 partitions resulted with the total weighted sum of error (WSE) also equals to 10. Since the matrix for the partitioning is small, we can also validate the clustering result manually. Compared with profile A and B, the only difference is that the access frequencies of query 1 and 2 are swapped. As a result, in order to minimize the total WSE in the profile matrix, there would be 3 partitions in the result for access profile A. However, for access profile B, the same set of partition arrangement could not be applied. As with the same arrangement, the total WSE would equal to 30. As a result, 4 partitions have to be resulted so that the total WSE would be reduced to 10.

4.5.2.4 Efficiency Analysis

In order to study the performance improvement in the partitioned collection of XML documents, we compare the time required for different set of queries. The set of queries are shown in Table 4-11. In the set of queries for experiment, we have used the same set of queries for partitioning so as to study the improvement of query processing after the ad hoc partitioning scheme is applied to the collection. For query 5, it involves all the partitions in the collection. This query can test the performance of the system when it has to search through the whole collection, i.e. it has to search through the whole set of partitions. For query 6, the query involves two

different partitions in order to get the results. This query is implemented to test the performance of the system when heterogeneous collection of XML documents is required to be the candidate of the results.

ID	XQuery expression
1	Same as those shown in Table 4-10
2	
3	
4	
5	for $x in collection('/db/Experiment/all') where $x//title &= "Database systems" return $x//title
6	for $x in collection('/db/Experiment/all') where $x//ProceedingsPage/location &= "San Francisco" or $x//IndexTermsPage//initPage < 40 return $x

Table 4-11 Set of queries for studying the efficiency of the partitioning scheme

To study the performance, we compared the computational time required for getting the result based on the set of queries described above. In fact, the computational time evaluated should consist of two components:

1. Time for determining the partitions involved (Query Manager)
2. Time for querying the expression in database system

In the experiments, we have found that the time required in the query manager is shorter than 1 millisecond (see Appendix II for the index tables used for query manager in Profile A and B). Since the time required is negligible, we compare the performance by the total time required for both components. Also, since the database

system has caching function, we have to restart the database for each query in different collections. The computation times required for each query in the un-partitioned collection, partition scheme based on profile A as well as profile B are shown in Table 4-12. The time elapsed are measure in millisecond. The improvement ratios compared with the un-partitioned collection are also evaluated for the partitioning scheme A and B as shown in Table 4-13.

	Query					
Collection	#1	#2	#3	#4	#5	#6
RAW[3]	797	734	984	765	844	968
A[4]	797	703	891	594	843	984
B[5]	781	719	891	594	844	1175

Table 4-12 The computation time for the set of queries in different collection (in ms)

	Query					
Collection	#1	#2	#3	#4	#5	#6
A	0.0%	5.4%	9.5%	22.3%	0.1%	-1.6%
B	2.0%	2.0%	9.5%	22.3%	0.0%	-21.3%

Table 4-13 The improvement ratio for the set of queries in different collection (in ms)

Refer to Table 4-13, the partitioned collection could improve the efficiency for the query from the set of user transaction patterns in general. However, it suffers from longer query time for the query which involves the whole collection of the XML documents, as the query engine requires time to process the set of partitions instead of one collection with the whole set of XML documents. The situation is even worse for the partitioning scheme B as the more the number of partitions, the much more time is required for processing. Consider the query #5, although the irrelevant set of

[3] RAW: un-partitioned collection of XML documents

[4] A: partitioned collection based on the access profile A

[5] B: partitioned collection based on the access profile B

XML documents are reduced, the time spent in processing heterogeneous collection of data have reduced the effect on the partitioning scheme. In overall, the efficiency of the query in the partitioned collection depends on two factors:

1. *The portion of irrelevant data reduced*

   In general, the more the number of irrelevant data reduced, the greater is the efficiency of the query. However, when considering query #1 in the partitioning A, it contradicts the situation as it could not get any improvement even though the query is based on the partitioned collection. In fact, comparing the query #1 with partitioning A and partitioning B, since the former one requires a large collection which is 93% of the overall XML documents collection to be looked-up, it could not get any improvement in the partitioned case. On the other hand, for the partitioning scheme B, the partition involved has only 44% of the overall documents in the collection; it could result in the improvement of the query processing.

2. *The number of partitions involved in the query*

   Since the query engine requires time for determining the location of the partitioned collection, the more the number of partitions involved, the more time is spent on looking up the partitions. As a result, as seen in the results of query #6 for partitioning scheme A and B, B suffers from a longer time for getting the result.

Since the formation of the partitioning scheme is based on the access profile provided, we have to evaluate whether the partitioning scheme formed is actually suitable for the access profile. In order to evaluate the fitness, we calculate the total sum of weighted time required for each query, in which the weighting is based on

the access frequencies provided in the access profile. The objective could be formulated as follow:

As refer to Table 4-13, the performance is worse for the un-partitioned collection, we only consider the partitioning scheme A and B to see whether it is suitable to its corresponding access profile. The evaluation result is shown in Table 4-14.

Partition Scheme	Access Profile A	Access Profile B
A	**36,485ms**	38,365ms
B	36,805ms	**38,045ms**

Table 4-14 The weighted time for evaluation

From Table 4-14, we have seen that the time required for the whole cycle of access transaction patterns is shorter when partitioning schemes are applied to their corresponding access profile. In this study, we can see that the partitioning scheme is ad hoc to the partitioning profile.

4.6 Chapter conclusion

In this section, we have performed two different set of experiments, one is based on clustering XML documents by structure and the other is based on clustering XML documents by access patterns. From the study, we can see that the BTACO-based block clustering could help us in providing a more accurate clustering analysis in identification of the XML schemas in the heterogeneous collection of XML documents. Even though it could not perform as effective as the S-GRACE method, the computational time is acceptable for the partitioning problem as it is usually as an off-line process. On the other hand, the high accuracy of identifying the groups of similar XML documents by structure can result in more generalized schemas for the XML documents and hence the query engine can avoid the irrelevant data in a more efficiency way.

In the experimental study, we can see that the database system with partitioned collection works better for the query set considered in the clustering process. This shows that the clustering method can partition the collections well for improving the specific set of query. Although the database system works worse for the other queries not in the user patterns, Navathe et al[88] stated that by the principle of 80 / 20 rule, 20% of the queries occupied 80% of the time for the database system. If we could improve the efficiency of those queries, the performance of the database system could still be improved in overall. This statement can also be applied in the case of XML database system. Although we have used eXist[1] as the platform for our experiments, the partitioning approach in fact could be implemented to other database system as discussed before.

The structure of XML documents is in fact very complicated and so sophisticated algorithm may be required for manipulating the XML collection. However, thorough consideration of the structure would suffer from high computational time complexity which is not an efficient and scalable for the real life. As a result, our proposed XML documents clustering approach would be appropriate as it is capable of dealing with the large dataset within an acceptable time.

Chapter 5 Conclusions

In this paper, we have proposed a modified version of Ant Colony Optimization algorithm, called Bi-tour Ant Colony Optimization (BTACO) method, to perform the block clustering. BTACO inherits the properties of ACO algorithms in combining heuristic search and learning effect in finding solution of a specific problem. To perform the block clustering, BTACO makes use of the distance measure and learning effect to find out a suitable partitioning scheme so as to reveal the clusters of objects and their associated attributes. The Weighted sum of error (WSE) is introduced as the goodness-of-fit of the partitioning scheme instead of using the mere distance measure as the measurement. To determine the partitioning scheme, we have proposed two partitioning approach, the dynamic programming approach and the heuristic partitioning approach, in our clustering method. The former one can find the better partitioning scheme but it suffers from a quadratic time complexity while the latter one can find the suitable partitioning scheme whenever the re-arranged data matrix is in a well-formed diagonal block structure. However, the heuristic partitioning approach can finish the search of partitioning scheme in linear time.

In our work, we have also introduced the use of BTACO clustering method in clustering the collection of XML documents. By clustering the XML documents, by structure, we can identify the groups of XML documents by structure and hence their schemas can be identified. The query engine can make use of the schemas in

avoiding searching the irrelevant data. We introduced BTACO clustering method to get a higher accuracy in identifying the groups of XML documents as it has a higher tolerance in dealing with the noisy data. Also, we improved the work of Lian et al[78] in evaluating the XML document similarity by introducing the weighing among the edges identified in the collection of XML documents.

Analogous to the vertical partitioning in the relational database design, we have introduced clustering the XML documents by the user transaction pattern. In our model, we have proposed using BTACO clustering method to cluster the XML documents into partitions. In order to query the partitioned collection, a query manager is designed which can determine the set of partitions required and can re-write the query expression whenever the query is inputted.

5.1 Contributions

Our proposed clustering method, Bi-tour Ant Colony Optimization (BTACO), works well in clustering the data matrix which is used for vertical partitioning problem. By the goodness-of-fit based on weighted sum of error (WSE) measurement, the partitioning schemes found are better than the schemes found by the Genetic Algorithm-based method and also the hierarchical clustering method. Also, our proposed method can provide a visualization of the relationship among the sets of attributes and the sets of transactions, which is an important issue in cluster analysis. Our method can provide a diagonal block structure of the re-arranged data matrix and so the clusters of the elements in the matrix are visualized as the sub-blocks along the diagonal line of the matrix. In the vertical partitioning in database design,

we can identify the outliers (inter-sub-block elements) in the database profile out easily and decide the further refinement of the database design.

From the computational study, we can see that BTACO clustering method can identify the groups of XML documents with similar structure more accurately, compared with methods using the edit-tree distance measure and common-edge distance measure respectively. The high accuracy is important in forming generalized schemas for the XML documents with similar structure. Also, in order to identify the schemas of the XML documents, we have modified the BTACO clustering method to support the classification and so as to cluster the XML documents with the defined number of clusters. The application use of BTACO clustering method becomes wider.

From the experiment in clustering XML documents by user transaction patterns, we can see that query in the partitioned collection can improve the efficiency of the query processing. Also, the BTACO clustering method provides a suitable partitioning scheme which is tailor-made to the specific user transaction profile in the XML documents. The efficiency of the query is greatly improved when the query requires only a very portion of XML documents. For the database system using partitioned collection, an efficient query manager is very important for determining the partitions required for query. Our proposed query manager can effectively identify the tag name together with their structural relationship by means of using region-based code in the schemas of the collection of XML documents which is better than the query manager proposed by Lian et at [78] as it requires the storage of the whole XML structures when dealing with relative path queries.

In fact, since the structural format of XML documents could be very complicated, sophisticated algorithm would be appropriate in dealing with the manipulation of the XML documents. However, the computational time required for such algorithm would be great as we have to consider the structure thoroughly. This is the case of using edit-tree distance measure approach for clustering XML documents by structure. Because of the time complexity problem, we have modified the common-edge approach which is an efficient and scalable approach in clustering the XML documents by structure.

In the same sense, for clustering XML documents by user access pattern, we have proposed the system which considers the ancestor-descendant relationship of the nodes in the XML documents only. Despite the lack of thorough consideration, improvement is made in the query process because of the two reasons: (1) the process of validating ancestor-descendant relationship could be finished in a very efficient way. (2) Validation of ancestor-descendant relationship can screen out the irrelevant data out of the time. The proposed clustering XML documents approach could be worthwhile in improvement of XML database design when the XML repositories are large.

5.2 Future works

In this paper, we can see that BTACO clustering method works well under the binary relation between the rows and columns of the data matrix. In fact, with the moderate modification of the method, the clustering method could also deals with

the matrix with real-valued data. By the capability of clustering real-valued data, the area of its application could be wider, as many problems are represented as real-valued matrices, such as many statistical data and gene expression data. The generalized version of BTACO could be more useful for the clustering analysis.

Refer to the conceptual lattice[84], the set of objects and sets of their associated attributes (descriptors) are clustered in a hierarchical way such that we can get a hierarchical relationship among the sets of objects and sets of attributes represented by a Hesse-like diagram. Such a clustering analysis is important in some applications like object-oriented database design. Recall back to our heuristic partitioning approach in BTACO clustering method, the approach finds the partitioning scheme by the principle of divide-and-conquer. Actually, the whole matrix can be regarded as the parent of the sub-matrices partitioned. With the moderate modification of BTACO clustering method, the hierarchical relationship among the sets of objects and the sets of attributes can also be revealed.

Bibliography

[1] "eXist - Open Source Native XML database," vol. 2005, 2005.

[2] "International Press Telecommunications Council, News Industry Text Format (NITF)," vol. 2005, 2000.

[3] "SIGMOD Record in XML," vol. 2005, 2002.

[4] Amir, Ben-Dor, Shamir Ron, and Yakhini Zohar, "Clustering Gene Expression Patterns," *Journal of Computational Biology*, vol. 6, pp. 281-297, 1999.

[5] Antonella, Carbonaro and Maniezzo Vittorio, "The Ant Colony Optimization paradigm for combinatorial optimization," in *Advances in evolutionary computing: theory and applications*: Springer-Verlag New York, Inc., 2003, pp. 539-557.

[6] Arabie, P., S. A. Boorman, and P. R. Levitt, "Constructing blockmodels: How and why"," *J. Mathemat. Psychol.*, vol. 17, pp. 21-63, 1978.

[7] Arahie, Phipps and Lawrence J. Hubert, "The Bond Energy Algorithm Revisited," *IEEE Trans. on Systems, Man, and Cybernetics*, vol. 20, pp. 268-274, 1990.

[8] Barbosa, Denilson, Alberto Mendelzon, John Keenleyside, and Kelly Lyons, "ToXgene: a template-based data generator for XML," in *Proceedings of the 2002 ACM SIGMOD international conference on Management of data*. Madison, Wisconsin: ACM Press, 2002.

[9] Beasley, David, David R. Bull, and Ralph R. Martin, "An Overview of Genetic Algorithms: Part 1, Fundamentals," *University Computing*, vol. 15, pp. 58-69, 1993.

[10] Beasley, David, David R. Bull, and Ralph R. Martin, "An overview of Genetic Algorithms: Part 2, Research Topics," *Unviersity Computing*, vol. 15, pp. 170-181, 1993.

[11] Beasly, David, David R. Bull, and Ralph R. Martin, "An overview of Genetic Algorithms: Part 2, Research Topics," *Unviersity Computing*, vol. 15, pp. 170-181, 1993.

[12] Berman, Leonard and Angel Diaz, "Data Descriptors by Example," vol. 2005,

1999.

[13] Bray, Tim, Jean Paoli, C. M. Sperberg-Mcqueen, Eve Maler, and François Yergeau, "Extensible Markup Language (XML) 1.0 (Third Edition)," vol. 2005, 2004.

[14] Bullnheimer, Bernd., Rickard F. Hartl, and Christine. Straus, "A new rank-based version of the ant system: a computational study," *Central European Journal Of Operations Research*, vol. 7, pp. 25-38, 1999.

[15] Bunke, Horst and Kim Shearer, "A graph distance metric based on the maximal common subgraph," *Pattern Recogn. Lett.*, vol. 19, pp. 255-259, 1998.

[16] Carpineto, C. and G Romano, "GALOIS: An order-theoretic approach to conceptual clustering," presented at Proceedings of 10th International Conference on Machine Learning, Amherst, 1993.

[17] Chawathe, S., "Comparing hierarchical data in extended memory," *In. Proc. of VLDB*, pp. 90-101, 1999.

[18] Chawathe, S., A. Rajaraman, H. Garcia-Molina, and J. Widom, "Change detection in hierarchically structured information," *In Proc. of ACM SIGMOD*, pp. 493-504, 1996.

[19] Chen, Q, A Lim, and K.W. Ong, "D(k)-index: An adaptive structural summary for graph-structured data," presented at SIGMOD, 2003.

[20] Chen, Yi, Susan B. Davidson, and Yifeng Zheng, "BLAS: an efficient XPath processing system," in *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*. Paris, France: ACM Press, 2004.

[21] Cheng, C. H., "A Branch and Bound Clustering Algorithm," *IEEE Trans. on Systems, Man, and Cybernetics*, vol. 25, pp. 895-898, 1995.

[22] Cheng, C. H., Y. P.; Gupta, W. H.; Lee, and K. F. Wong, "A TSP-based heuristic for forming machine groups and part families," *International Journal of Production Research*, vol. 36, pp. 1325-1337, 1998.

[23] Cheng, Chun-Hung, Wing-Kin Lee, and Kam-Fai Wong, "A genetic algorithm-based clustering approach for database partitioning," *Systems, Man and Cybernetics, Part C, IEEE Transactions on*, vol. 32, pp. 215-230, 2002.

[24] Cheng, Yizong and George M. Church, "Biclustering of Expression Data," in *Proceedings of the Eighth International Conference on Intelligent Systems for Molecular Biology*: AAAI Press, 2000.

[25] Clark, James and Steve Derose, "World Wide Web Consortium - XML Path Language (XPath) Version 1.0," vol. 2005, 1999.

[26] Codd, E.F., "Further normalization of the database relational model," *Database Systems, Courant Computer Science Symposia*, vol. 6, pp. 65-98, 1972.

[27] Codd, E.F., "Recent investigations in relational database systems," *Proc. 1974 IFIP Congr.*, pp. 397-434, 1974.

[28] Codd, E.F., "A Relational Model of Data for Large Shared Data Banks," *Communications of the ACM*, vol. 13, 1970.

[29] Cornell, Douglas W. and Philip S. Yu, "An Effective Approach to Vertical Partitioning for Physical Design of Relational Databases," *IEEE Trans. Softw. Eng.*, vol. 16, pp. 248-258, 1990.

[30] Cornell, Douglas W. and Philip S. Yu, "A Vertical Partitioning Algorithm for Relational Databases," in *Proceedings of the Third International Conference on Data Engineering*: IEEE Computer Society, 1987.

[31] Davey, B. and H. Priestley, *Introduction to Lattices and Order*: Cambridge University Press, Cambridge, Great Britain, 1990.

[32] Davidson, A., M. Fuchs, and M. Hedin, "Schema for Object-Oriented XML 2.0," vol. 2005, 1999.

[33] Diaz, A.L. and D. Lovell, "XML Generator," vol. 2005, 1999.

[34] Dorigo, M. and L. M. Gambardella, "Ant colony system: a cooperative learning approach to the traveling salesman problem," *Evolutionary Computation, IEEE Transactions on*, vol. 1, pp. 53-66, 1997.

[35] Dorigo, Marco, "Optimization, learning and natural algorithms," vol. Ph.D. Milano: Politecnico di Milano, 1992.

[36] Dorigo, Marco and Gianni Di Caro, "Ant Colony Optimization: A New Meta-Heuristic," presented at Proceedings of the Congress on Evolutionary Computation, 1999.

[37] Duffy, D. E. and A. J. Quiroz, "A Permutation-Based Algorithm for Block Clustering," *Journal of Classification*, vol. 8, pp. 65-91, 1991.

[38] Eve, Maler and El Andaloussi Jeanne, *Developing SGML DTDs: from text to model to markup*. Upper Saddle River, N.J.: Prentice Hall PTR, 1996.

[39] Fagin, Ronald, "Multivalues dependencies and a new normal form for relational databases," *ACM Trans. Database Syst.*, vol. 2, pp. 262-278, 1977.

[40] Fagin, Ronald, "Normal forms and relational database operators," *Proc. ACM SIGMOD*, pp. 153-160, 1979.

[41] Fern, Mirtha-Lina, Ndez, and Gabriel Valiente, "A graph distance metric combining maximum common subgraph and minimum common supergraph," *Pattern Recogn. Lett.*, vol. 22, pp. 753-758, 2001.

[42] Fisher, D, "Knowledge acquisition via incremental conceptual clustering," *Machine Learning*, vol. 2, pp. 139-172, 1987.

[43] Frankston, C. and H.S. Thompson, "XML-Data reduced," vol. 2005, 1998.

[44] Gambardella, L.M. and M. Dorigo, "Ant-Q: A reinforcement learning approach to the travelling salesman problem," presented at Proceedings of the twelfth International Conference on Machine Learning, ML-95, Palo Alto, CA, Morgan Kaufmann, Palo Alto, California, USA, 1995.

[45] Gambardella, L.M., E.D. Taillard, and M. Dorigo, "Ant colonies for the quadratic assignment problem," *Journal of the Operational Research Society*, vol. 50, pp. 167-176, 1999.

[46] Gambardella, Luca Maria and Marco Dorigo, "An Ant Colony System Hybridized with a New Local Search for the Sequential Ordering Problem," *INFORMS J. on Computing*, vol. 12, pp. 237-255, 2000.

[47] Garofalakis, M., A. Gionis, R. Rastogi, S. Seshadri, and K. Shim, "Xtract: A system for extracting document type descriptors from XML documents," *In Proc. of ACM SIGMOD*, pp. 165-176, 2000.

[48] Goldman, R. and J. Widom, "DataGuides: Enable query formulation and optimization in semistructured databases," presented at In VLDB, 1997.

[49] Govaert, Gerard and Mohamed Nadif, "Clustering with block mixture models," *Pattern Recognition*, vol. 36, pp. 463-473, 2003.

[50] Grust, Torsten, "Accelerating XPath location steps," in *Proceedings of the 2002 ACM SIGMOD international conference on Management of data*. Madison, Wisconsin: ACM Press, 2002.

[51] Guha, S., R. Rastogi, and K. Shim, "ROCK: A Robust Clustering Algorithm For Categorical Attributes," presented at Proc. 15th Int'l Conf. Data Eng., 1999.

[52] Hadzikadic, M. and D. Yun, "Concept formation by incremental conceptual clustering," presented at Proceedings of the 11th International Joint Conference on Artificial Intelligence, Detroit, MI: Morgan Kaufmann, 1989.

[53] Hanson, S. and M Bauer, "Conceptual clustering, categorization, and polymorphy," *Machine Learning*, pp. 343-372, 1989.

[54] Hartigan, J.A., "Direct clustering of a data matrix," *Journal of the American Statistical Association*, vol. 67, pp. 123-129, 1972.

[55] Hartigan, John A., *Clustering algorithms*: Wiley, 1975.

[56] He, H. and J. Yang, "Multiresolution indexing of XML for frequent queries," presented at ICDE, 2004.

[57] Hegaret, Philippe Le, Ray Whitmer, and Lauren Wood, "World Wide Web

Consortium - Document Object Model," vol. 2005, 2005.

[58] Hencscy, G., "Associative-structural analysis of 0-1 data using montone systems," *Automat. Remote Contr.*, vol. 48, pp. 247-250, 1987.

[59] Heragu, S. S., "Group technology and cellular manufacturing," *Systems, Man and Cybernetics, IEEE Transactions on*, vol. 24, pp. 203-215, 1994.

[60] Hoffer, J.A., "An integer programming formulation of computer database design problems," *Information Sicences*, vol. 11, pp. 29-48, 1976.

[61] Hu, Yanhai, Feifan Ye, and Zhimei Fang, "A study on the integration of lean production and group technology," 2000.

[62] Hubert, L. J. and F. B. Baker, "Application of combinatorial programming to data analysis: The traveling salesman and related problems," vol. 43, pp. 81-91, 1978.

[63] Jain, A. K., M. N. Murty, and P. J. Flynn, "Data clustering: a review," *ACM Comput. Surv.*, vol. 31, pp. 264-323, 1999.

[64] Jelliffe, R., "Schematron," vol. 2005, 2000.

[65] Kaelbling, Leslie Pack, Michael L. Littman, and Andrew W. Moore, "Reinforcement Learning: A Survey," *Journal of Artificial Intelligence Research*, vol. 4, pp. 237-285, 1996.

[66] Kalkidi, Maria, Yannis Batistakis, and Michalis Vazirgannis, "On Clustering Validation Techniques," *Journal of Intelligent Information Systems*, vol. 17, pp. 107-145, 2001.

[67] Kaushik, R., P. Sheony, P. Bohannon, and E. Gudes, "Exploiting local similarity for efficient indexing of paths in graph structured data," presented at ICDE, 2002.

[68] Kha, D. D., M. Yoshikawa, and S. Uemura, "An XML indexing structure with relative region coordinate," 2001.

[69] King, J. R., "Machine-component grouping in production flow analysis: An approach using a rank order clustering algorithm," *International Journal of Production Research*, vol. 18, pp. 213-232, 1980.

[70] Klarlund, N., A. Moller, and M. I. Schwatzbach, "DSD: A Schema Language for XML," presented at 3rd ACM Workshop on Formal Methods in Software Practice, 2000.

[71] Kusiak, A., A. Vannelli, and K.R. Kumar, "Clustering analysis: Models and algorithms," *Contr. Cybern.*, vol. 15, pp. 139-154, 1986.

[72] Kusiak, Andrew and Chun Hung Cheng, "A branch-and-bound algorithm for solving the group technology problem," *Ann. Oper. Res.*, vol. 26, pp. 415-431, 1990.

[73] Kusiak, Andrew and W.S. Chow, "Efficient solving of the group technology problem," *International Journal of Production Research*, vol. 25, 1987.

[74] Lebowitz, M., *Concept learning in a rich input domain: generalization-based memory.*, vol. 2: San Mateo, CA: Morgan Kaufmann, 1986.

[75] Lee, Mong Li, Liang Huai Yang, Wynne Hsu, and Xai Yang, "XClust: clustering XML schemas for effective integration," presented at Proceedings of the eleventh international conference on Information and knowledge management, McLean, Virginia, USA, 2002.

[76] Lenstra, J.K. and A.H.G. Kan Rinnoy, "Some application of the traveling salesman problem," *Oper. Res. Q.*, vol. 26, pp. 717-733, 1975.

[77] Li, Quanzhong and Bongki Moon, "Indexing and Querying XML Data for Regular Path Expressions," in *Proceedings of the 27th International Conference on Very Large Data Bases*: Morgan Kaufmann Publishers Inc., 2001.

[78] Lian, Wang, David Wai-Lok Cheung, Nikos Mamoulis, and Siu-Ming Yiu, "An Efficient and Scalable Algorithm for Clustering XML Documents by Structure," *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, pp. 82-96, 2004.

[79] Lin, X. and Y. Zhang, "A new graphical method for vertical partitioning in distributed database design," presented at Proc. 4th Australian Database Conf., 1993.

[80] Marcotorchino, F., "Block seriation problems: A unified approach," *Appl. Stochastic Models and Data Anal*, vol. 3, pp. 73-91, 1987.

[81] Martin, J. and Billman D., "Acquiring and combining overlapping concepts.," *Machine Learning*, vol. 16, pp. 121-155, 1994.

[82] Mccormick, J.W.T., P. J. Schweizer, and T.W. White, "Problem decomposition and data reorganization by a clustering technique," *Oper. Res.*, vol. 20, pp. 993-1009, 1972.

[83] Michael, J. Berry and Linoff Gordon, *Data Mining Techniques: For Marketing, Sales, and Customer Support*: John Wiley \& Sons, Inc., 1997.

[84] Michalski, R.S and R. Stepp, "Learning from observation: conceptual clustering," *Machine Learning: An Artificial Intelligence Approach (Vol.1)*, 1983.

[85] Milo, T. and D. Suciu, "Index structures for path expression," presented at In Proceedings of 7th International Conference on Database Theory, 1999.

[86] Mirkin, B. G., *Mathematical classification and clustering*. Dordrecht; Boston:

Kluwer Academic Pubishers, 1996.

[87] Navathe, Shamkant B. and Mingyoung Ra, "Vertical partitioning for database design: a graphical algorithm," in *Proceedings of the 1989 ACM SIGMOD international conference on Management of data*. Portland, Oregon, United States: ACM Press, 1989.

[88] Navathe, Shamkant, Stefano Ceri, Gio Wiederhold, and Jinglie Dou, "Vertical partitioning algorithms for database design," *ACM Trans. Database Syst.*, vol. 9, pp. 680-710, 1984.

[89] Nierman, A. and H.V. Jagadish, "Evaluating Structural Similarity in XML Documents," *Proc. Fifth Int'l Workshop Web and Databases*, 2002.

[90] Quin, Liam, "World Wide Web Consortium - XML," vol. 2005, 2003.

[91] Quin, Liam, "World Wide Web Consortium - XML Query," vol. 2005, 2005.

[92] Reeves, Colin R., *Modern heuristic techniques for combinatorial problems*. New York: Halsted Press, 1993.

[93] Rutenbar, R. A., "Simulated annealing algorithms: an overview," *Circuits and Devices Magazine, IEEE*, vol. 5, pp. 19-26, 1989.

[94] Shanmugasundaram, J., K. Tufte, G. He, C. Zhang, D. Dewitt, and J. Naughton, "Relational Databases for Querying XML Documents: Limitations and Opportunties," *Proc. 25th Int'l Conf. Very Large Data Bases*, pp. 302-314, 1999.

[95] Silberschatz, Abraham, Henry F. Korth, and S. Sudarshan, *Database system concepts*, 4th ed. Boston: McGraw-Hill, 2002.

[96] Spaeth, H., *Cluster Analysis Algorithmss, (U. Bull. trans.)*, vol. 1980. Chichester, England: Ellis Horwood, 1980.

[97] Sperberg-Mcqueen, C.M. and Henry Thompson, "World Wide Web Consortium - XML Schema," vol. 2005, 2001.

[98] St. Laurent, Simon and Robert Biggar, "Inside XML DTDs." New York: McGraw-Hill, 1999.

[99] Tatarinov, Igor, Stratis D. Viglas, Kevin Beyer, Jayavel Shanmugasundaram, Eugene Shekita, and Chun Zhang, "Storing and querying ordered XML using a relational database system," in *Proceedings of the 2002 ACM SIGMOD international conference on Management of data*. Madison, Wisconsin: ACM Press, 2002.

[100] Wang, Haixun, Wei Wang, Jiong Yang, and Philip S. Yu, "Clustering by pattern similarity in large data sets," in *Proceedings of the 2002 ACM SIGMOD international conference on Management of data*. Madison, Wisconsin: ACM Press, 2002.

[101] Wang, Wei, Haifeng Jiang, Hongjun Lu, and Jeffrey Xu Yu, "PBiTree Coding and Efficient Processing of Containment Joins," presented at In Proc. 19th International Conference on Data Engineering, 2003.

[102] Whitten, Jeffrey L., Lonnie D. Bentley, and Kevin C. Dittman, *Systems analysis and design methods*, 5th ed. Boston: McGraw-Hill/Irwin, 2000.

[103] Witten, I. H. and Eibe Frank, *Data mining: practical machine learning tools and techniques with Java implementations*: Morgan Kaufmann, 2000.

[104] Woo, K.H. and C.H Cheng, "Bi-Tour Ant Colony Optimization for Diagonal Clustering," presented at The 4th IEEE Workshop of Soft Computing as Transciplinary and Science and Technology, Muroran, Japan, 2005.

[105] Yang, J., W. Wang, H. Wang, and P.S. Yu, "δ-clusters: Capturing subspace correlation in a large data set," presented at ICDE, 2002.

[106] Zhang, Chun, Jeffrey Naughton, David Dewitt, Qiong Luo, and Guy Lohman, "On supporting containment queries in relational database management systems," in *Proceedings of the 2001 ACM SIGMOD international conference on Management of data*. Santa Barbara, California, United States: ACM Press, 2001.

[107] Zhang, K. and D. Shasha, "Simple fast algorithms for the editing distance between trees and related problems," *SIAM Journal of Computing*, vol. 18, pp. 1245-1262, 1989.

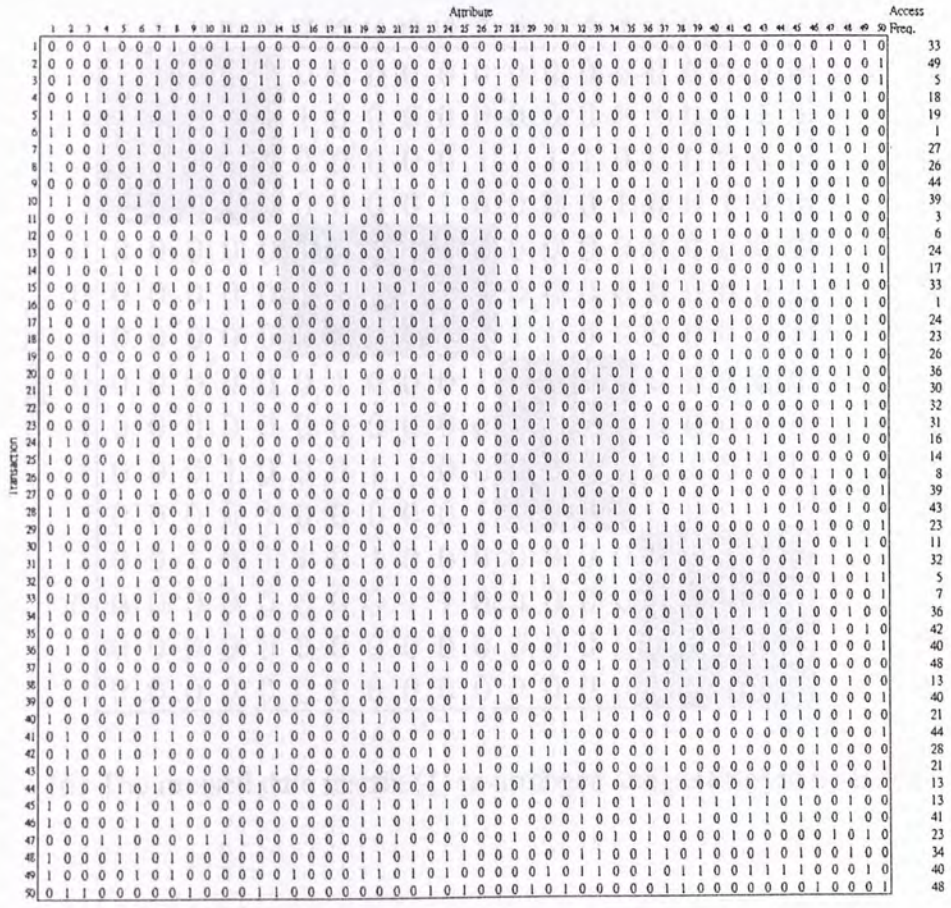
Appendix I

Transaction	Attributes																				Access
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	Freq.
1	1	0	0	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	50
2	0	1	0	0	0	0	0	0	1	0	0	1	1	1	0	0	0	0	0	0	50
3	0	0	1	0	0	0	1	0	0	1	1	0	0	0	0	0	1	1	0	0	50
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	1	50
5	1	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	15
6	1	0	0	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	0	15
7	0	0	1	0	0	0	1	0	0	1	1	0	0	0	0	0	1	0	0	0	15
8	0	1	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	1	0	1	15
9	0	1	0	0	1	0	0	0	0	0	1	0	0	1	0	0	0	0	1	0	10
10	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	1	0	0	10
11	1	1	1	0	1	1	0	0	1	0	0	1	1	0	0	0	0	0	0	0	10
12	0	0	0	1	0	0	1	0	0	1	0	0	0	1	0	0	0	0	1	1	10
13	0	0	0	0	0	0	0	1	0	0	1	0	0	0	1	1	1	1	0	0	10
14	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	5
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	5

(a) The original matrix for **data matrix 1**

Transaction	Attribute																																		Access
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	Freq.
1	0	1	0	0	1	0	1	0	1	0	0	0	1	0	0	0	1	1	0	0	0	1	0	1	1	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	13
2	1	1	0	0	0	0	1	0	0	1	1	0	0	0	0	0	1	1	0	1	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	1	0	22
3	1	0	0	1	0	1	0	1	1	1	0	1	1	0	0	0	0	1	1	0	1	1	0	1	1	0	0	0	1	0	0	0	0	1	0	1	1	0	1	0	8
4	0	1	1	0	1	1	1	1	0	1	0	1	0	0	0	0	0	0	1	0	0	1	1	0	0	1	0	0	1	0	0	0	0	1	0	1	0	0	1	0	32
5	0	1	0	1	1	1	1	1	0	0	1	0	0	0	0	1	1	1	0	0	1	1	0	0	0	1	0	0	1	1	1	0	1	0	0	1	0	0	0	1	24
6	1	1	1	1	0	0	1	1	0	1	0	1	0	0	0	1	1	1	0	0	0	0	0	1	1	1	0	1	1	0	1	0	0	1	0	1	0	0	1	0	30
7	0	1	0	1	1	0	1	0	1	0	0	0	0	0	0	1	0	0	0	0	0	1	1	0	1	0	1	1	0	0	1	0	1	1	0	1	1	1	0	1	42
8	1	0	1	1	0	0	0	0	1	0	0	1	0	1	0	1	1	0	0	0	1	1	0	0	1	1	0	0	0	1	0	1	0	0	1	0	1	1	1	1	22
9	0	1	0	1	0	0	0	0	1	0	0	0	0	0	1	0	0	0	1	0	0	1	0	0	1	1	1	0	0	0	0	1	1	0	0	0	0	0	0	0	8
10	1	1	0	1	1	0	0	0	1	0	0	1	0	0	0	0	1	0	1	1	1	1	0	0	1	0	0	1	0	1	1	1	0	0	1	1	0	0	0	1	22
11	0	0	1	1	0	0	1	1	1	1	0	0	1	1	0	0	0	1	1	1	0	0	0	0	0	1	1	0	1	1	0	0	1	1	1	1	0	1	0	0	24
12	1	0	1	0	0	1	0	1	0	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	0	0	1	1	1	0	1	1	0	1	0	1	0	0	1	0	33
13	0	0	0	0	1	0	1	0	0	0	1	0	1	0	0	0	1	0	0	0	1	0	0	0	0	0	1	0	0	1	0	1	0	1	0	1	1	0	0	0	50
14	0	0	1	1	1	0	0	0	0	0	0	0	1	0	1	1	0	0	0	1	1	0	1	1	1	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	28
15	0	0	1	0	0	0	1	0	0	1	0	0	0	1	0	0	0	1	1	1	0	0	0	0	0	0	1	1	0	1	0	0	0	0	1	0	1	1	0	1	22

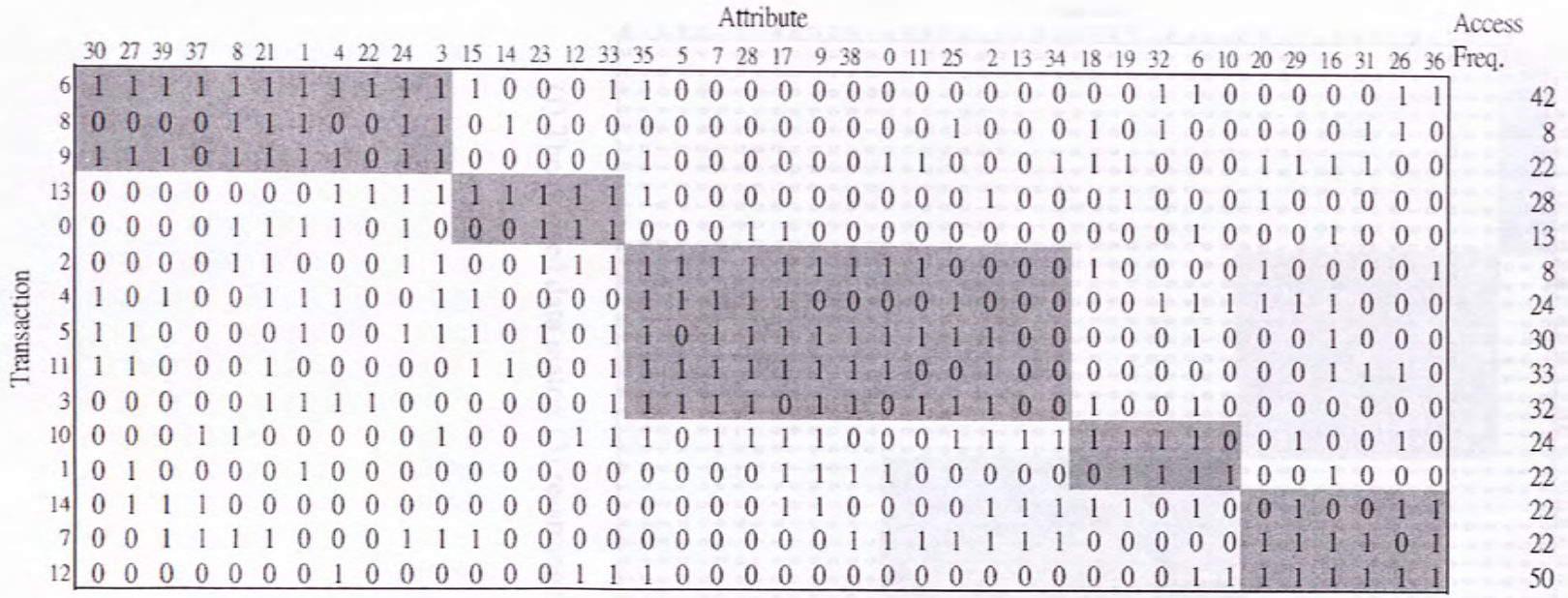
(b) The original data matrix for **data matrix 2**



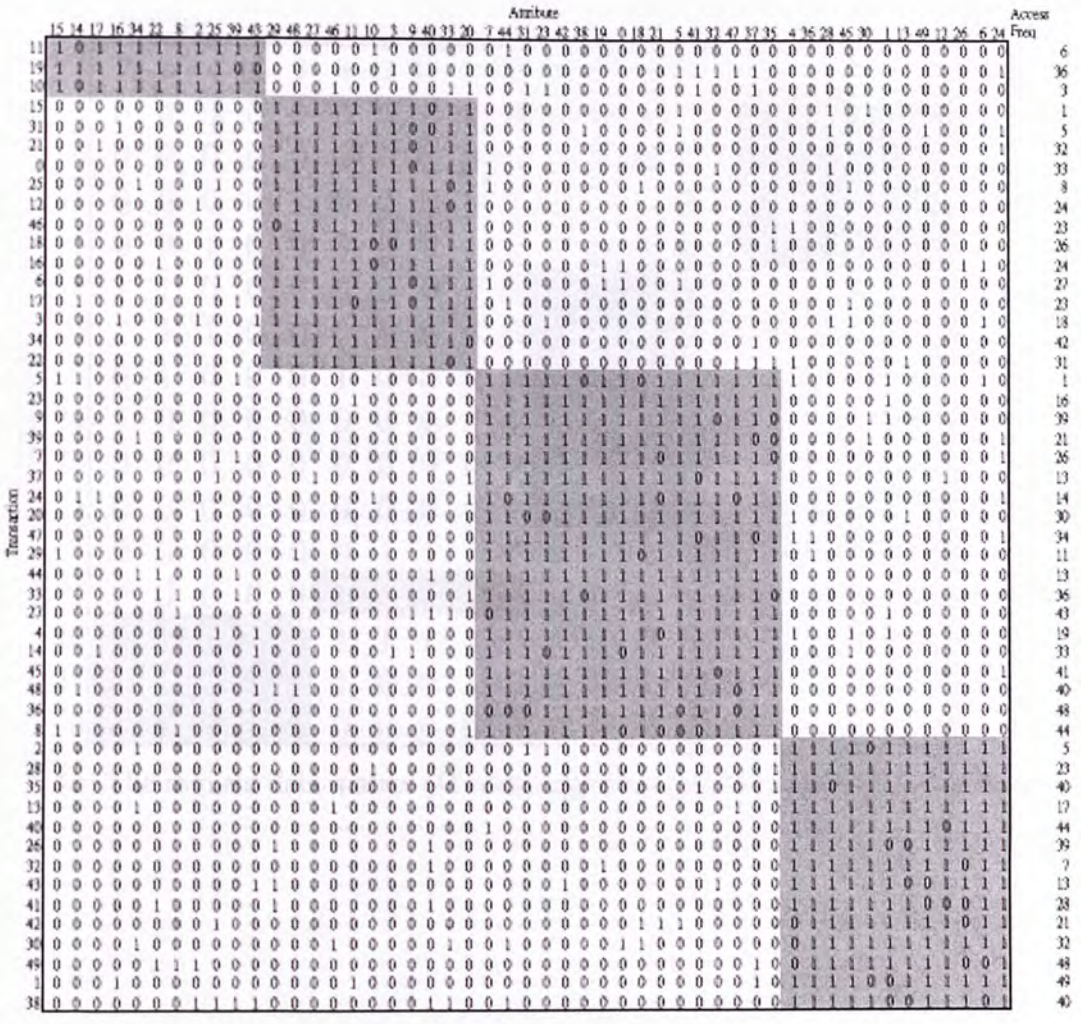
(c) The original matrix for **data matrix 3**

Transaction	Attribute																				Access
	2	6	9	10	16	17	15	14	19	18	3	5	4	0	7	8	11	12	1	13	freq.
6	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	50
2	1	1	1	0	1	1	0	0	0	0	0	0	0	0	0	1	0	1	1	0	10
12	1	1	1	1	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	15
3	0	0	1	1	0	0	0	0	0	1	0	0	0	0	1	0	0	1	0	0	10
11	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	15
14	0	0	0	0	0	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	50
9	0	0	0	0	0	0	1	0	0	1	1	1	1	0	0	0	0	0	0	1	10
13	0	0	0	0	0	0	1	0	0	0	1	1	1	1	1	0	0	0	0	0	5
0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	50
4	0	0	0	1	0	0	0	1	1	0	0	0	0	1	1	0	1	0	0	0	10
5	0	0	0	0	1	0	0	0	0	0	1	1	0	0	0	0	0	0	1	0	10
8	0	0	1	0	1	1	0	1	0	0	0	0	0	0	0	1	1	1	1	1	5
10	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	1	1	15
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	50
7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	15

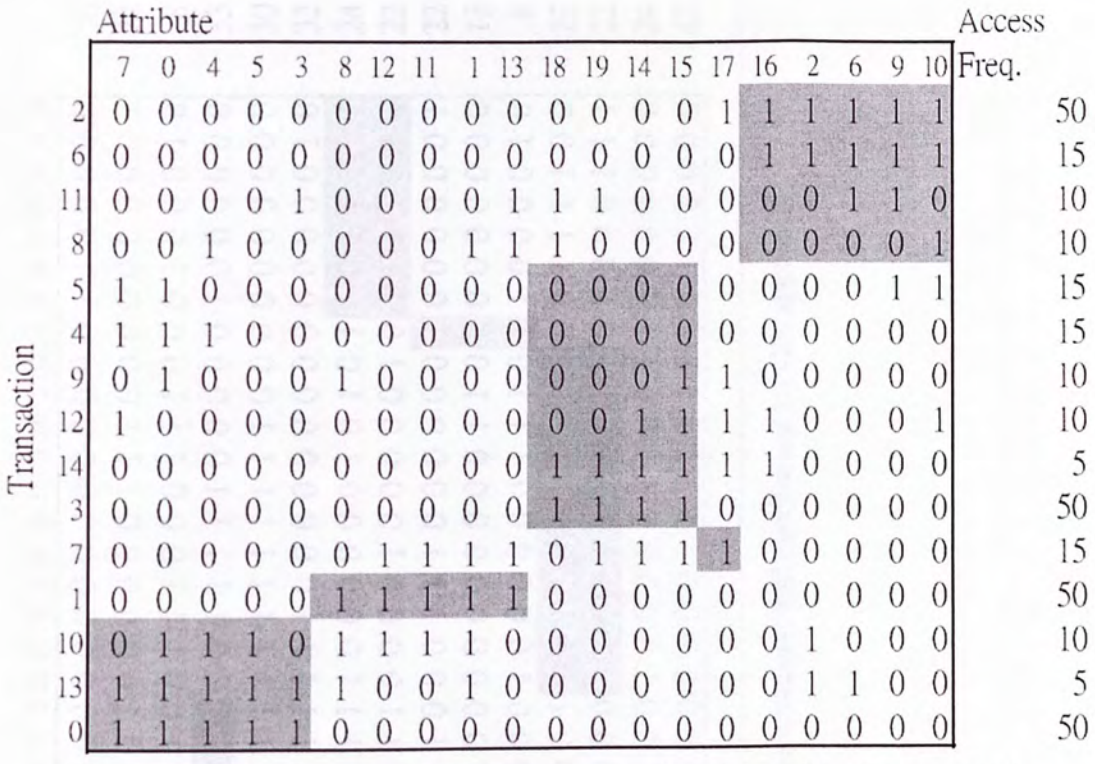
(d) The revised data matrix (1) re-arranged and partitioned by BTACO



(e) The revised data matrix (2) re-arranged and partitioned by BTACO

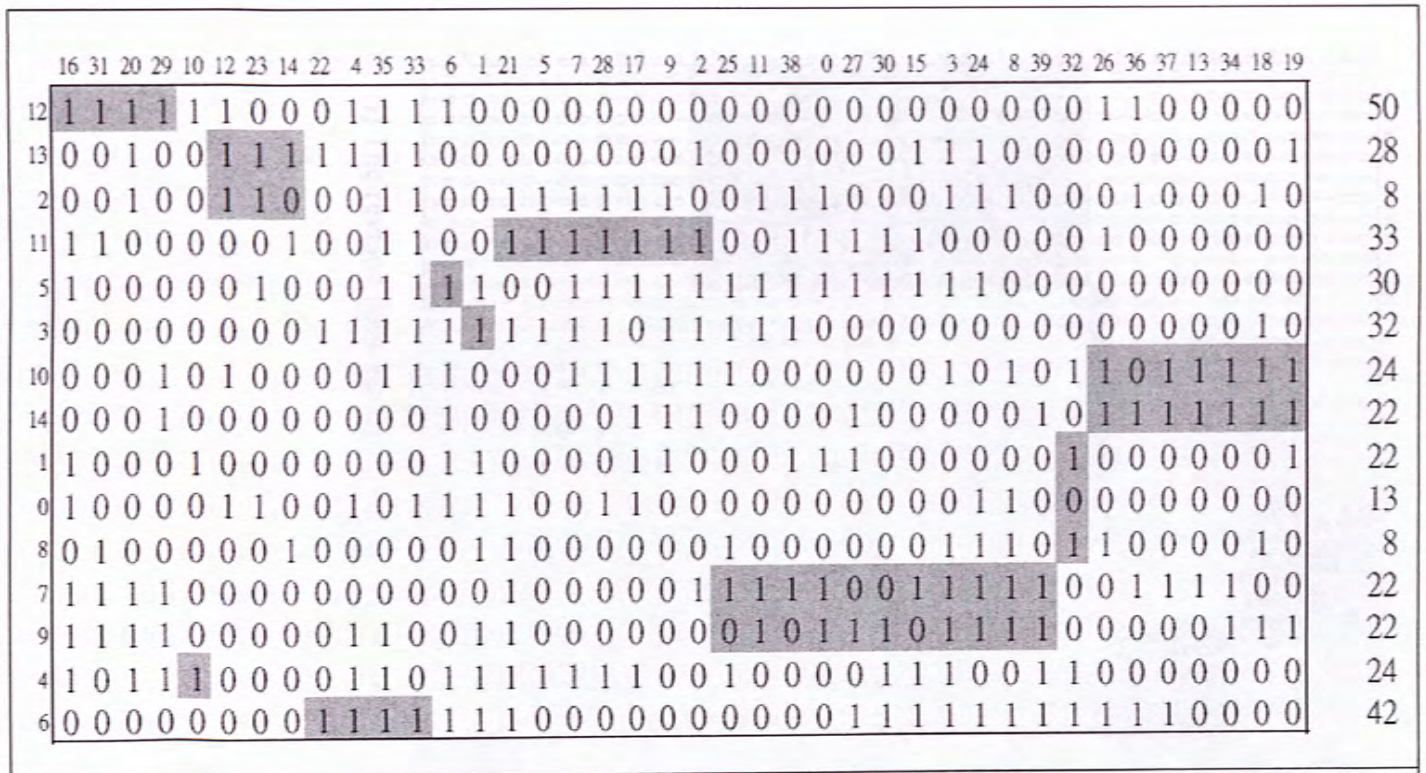


(f) The revised data matrix (3) re-arranged and partitioned by BTACO

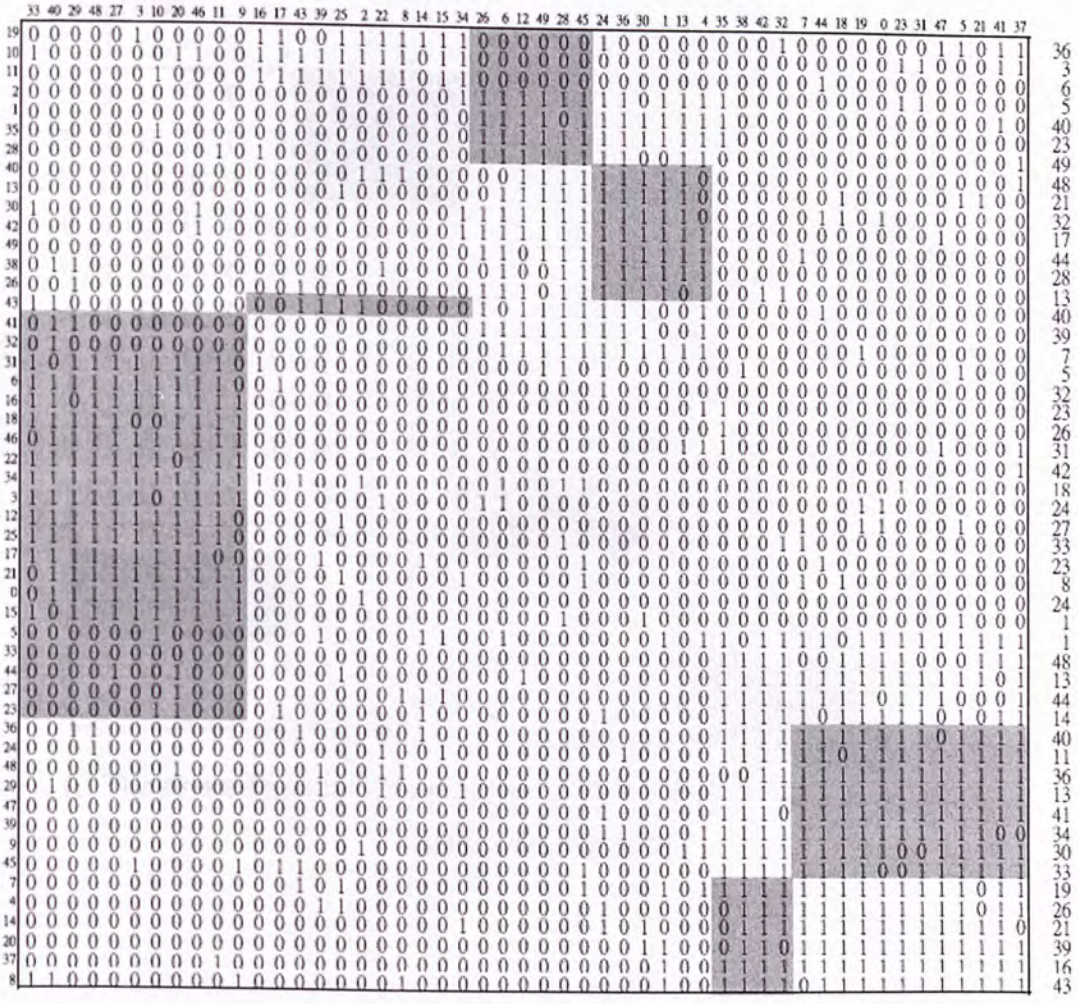


Transaction	Attribute																				Access Freq.
	7	0	4	5	3	8	12	11	1	13	18	19	14	15	17	16	2	6	9	10	
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	50
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	15
11	0	0	0	0	1	0	0	0	0	1	1	1	0	0	0	0	0	1	1	0	10
8	0	0	1	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	1	10
5	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	15
4	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	15
9	0	1	0	0	0	1	0	0	0	0	0	0	0	1	1	0	0	0	0	0	10
12	1	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	1	10
14	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	0	0	0	0	5
3	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	50
7	0	0	0	0	0	0	1	1	1	1	0	1	1	1	1	0	0	0	0	0	15
1	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	50
10	0	1	1	1	0	1	1	1	1	0	0	0	0	0	0	0	1	0	0	0	10
13	1	1	1	1	1	1	0	0	1	0	0	0	0	0	0	0	1	1	0	0	5
0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	50

(g) The revised data matrix (1) re-arranged and partitioned by GA approach



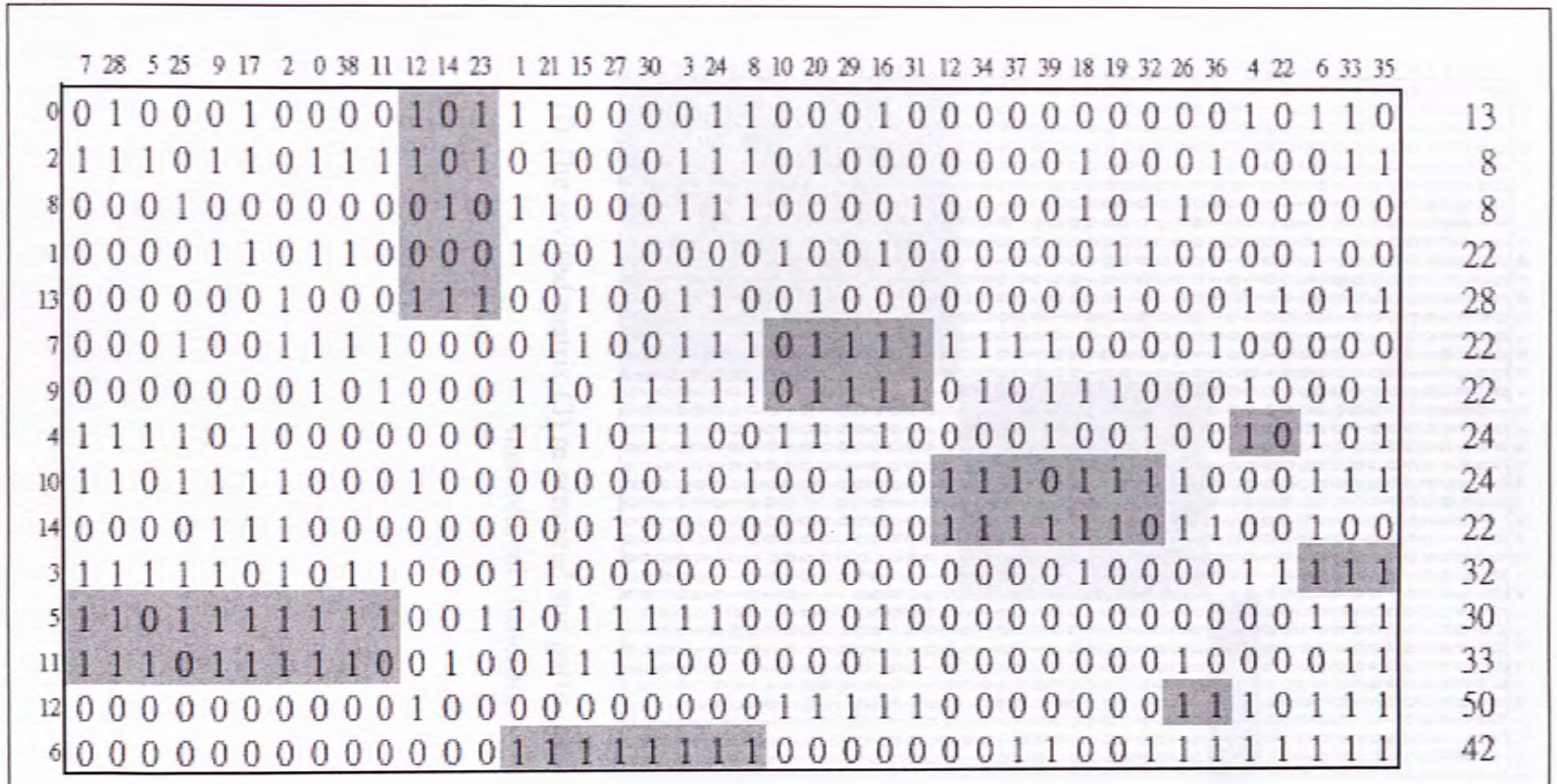
(h) The revised data matrix (2) re-arranged and partitioned by GA approach



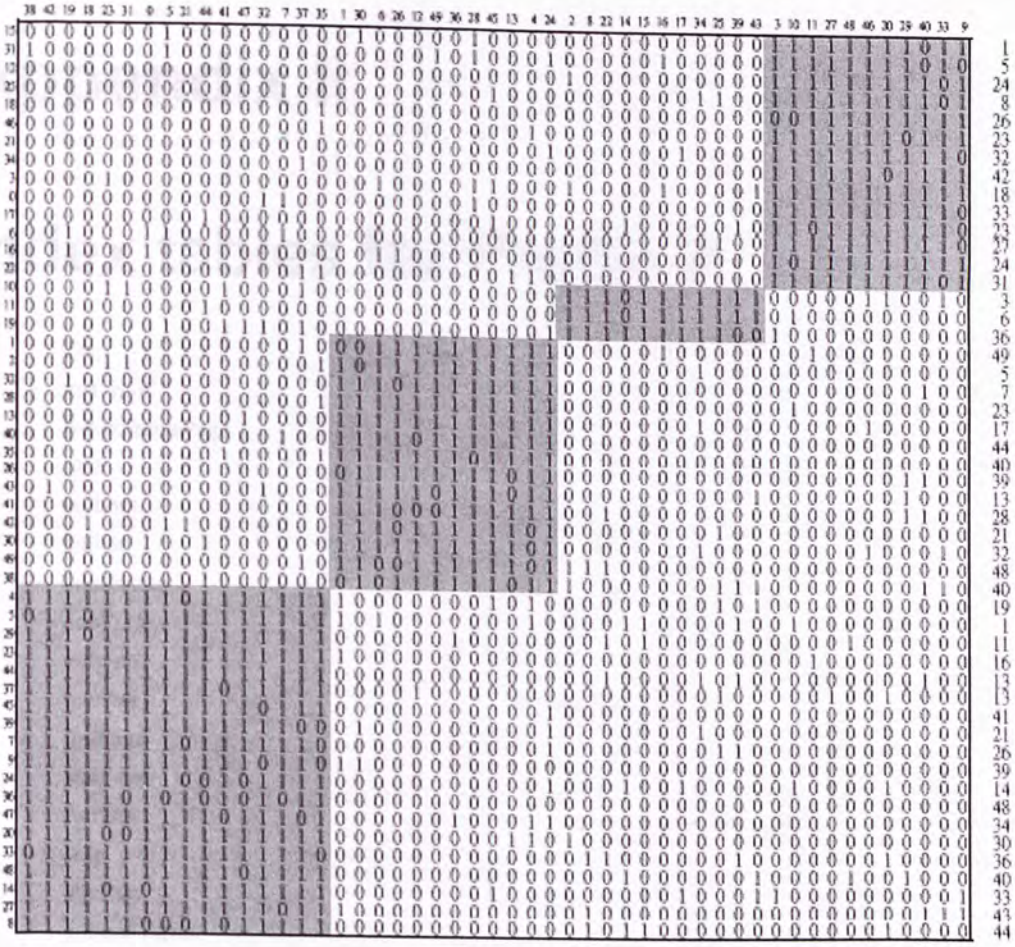
(i) The revised data matrix (3) re-arranged and partitioned by GA approach

Transaction \ Attribute	11	12	1	13	8	14	15	18	19	0	7	3	5	4	2	16	6	9	10	17	Access Freq.
3	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	50
14	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	1	0	0	0	1	5
7	1	1	1	1	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0	1	15
4	0	0	0	0	0	0	0	0	0	1	1	0	0	1	0	0	0	0	0	0	15
5	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1	1	0	15
9	0	0	0	0	1	0	1	0	0	1	0	0	0	0	0	0	0	0	0	1	10
12	0	0	0	0	0	1	1	0	0	0	1	0	0	0	0	1	0	0	1	1	10
8	0	0	1	1	0	0	0	1	0	0	0	0	0	1	0	0	0	0	1	0	10
11	0	0	0	1	0	0	0	1	1	0	0	1	0	0	0	0	1	1	0	0	10
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0	50
13	0	0	1	0	1	0	0	0	0	1	1	1	1	1	1	0	1	0	0	0	5
1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	50
10	1	1	1	0	1	0	0	0	0	1	0	0	1	1	1	0	0	0	0	0	10
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	50
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	15

(j) The revised matrix (1) re-arranged and partitioned by average-linkage hierarchical clustering



(k) The revised matrix (2) re-arranged and partitioned by average-linkage hierarchical clustering



(l) The revised matrix (3) re-arranged and partitioned by average-linkage hierarchical clustering

Appendix II

Index tables for Profile A

Tag	Start	End	Level
virt_root	1	40	0
IndexTermsPage	2	39	1
title	3	4	2
authors	5	8	2
author	6	7	3
confName	9	10	2
confYear	11	12	2
volume	13	14	2
number	15	16	2
initPage	17	18	2
endPage	19	20	2
fullText	21	24	2
size	22	23	3
abstract	25	26	2
generalTerms	27	30	2
term	28	29	3
categoryAndSubjectDescriptors	31	38	2
categoryAndSubjectDescriptorsTuple	32	37	3
category	33	34	4
content	35	36	4

Index for partition 1

Tag	Start	End	Level
virt_root	1	80	0
OrdinaryIssuePage	2	33	1
volume	3	4	2
number	5	6	2
month	7	8	2
year	9	10	2
sectionList	11	32	2
sectionListTuple	12	31	3
sectionName	13	14	4
articles	15	30	4
articlesTuple	16	29	5
toArticle	17	20	6
title	18	19	7
initPage	21	22	6
endPage	23	24	6
authors	25	28	6
author	26	27	7
ProceedingsPage	34	79	1
volume	35	36	2
number	37	38	2
month	39	40	2
year	41	42	2
conference	43	44	2
date	45	46	2
confyear	47	48	2
location	49	50	2
sectionList	51	78	2
sectionListTuple	52	77	3
sectionName	53	54	4
articles	55	76	4
articlesTuple	56	75	5
title	57	58	6
authors	59	62	6
author	60	61	7
initPage	63	64	6
endPage	65	66	6
Toindex	67	70	6
index	68	69	7
fullText	71	74	6
size	72	73	7

Index table for partition 2

Tag	Start	End	Level
virt_root	1	24	0
SigmodRecord	2	23	1
issue	3	22	2
volume	4	5	3
number	6	7	3
articles	8	21	3
article	9	20	4
title	10	11	5
initPage	12	13	5
endPage	14	15	5
authors	16	19	5
author	17	18	6

Index table for partition 3

Index tables for Profile B

Tag		Start	End	Level
virt root	1	32	0	
IndexTermsPage	2	31	1	
title	3	4	2	
authors	5	8	2	
author	6	7	3	
confName	9	10	2	
confYear	11	12	2	
volume	13	14	2	
number	15	16	2	
initPage	17	18	2	
endPage	19	20	2	
fullText	21	24	2	
size	22	23	3	
abstract	25	26	2	
generalTerms	27	28	2	
categoryAndSubjectDescriptors	29	30	2	

Index table for partition 1

Tag		Start	End	Level
virt root	1	40	0	
IndexTermsPage	2	39	1	
title	3	4	2	
authors	5	8	2	
author	6	7	3	
confName	9	10	2	
confYear	11	12	2	
volume	13	14	2	
number	15	16	2	
initPage	17	18	2	
endPage	19	20	2	
fullText	21	24	2	
size	22	23	3	
abstract	25	26	2	
generalTerms	27	30	2	
term	28	29	3	
categoryAndSubjectDescriptors	31	38	2	
categoryAndSubjectDescriptorsTuple	32	37	3	
category	33	34	4	
content	35	36	4	

Index table for partition 2

Tag	Start	End	Level
virt_root	1	80	0
OrdinaryIssuePage	2	33	1
volume	3	4	2
number	5	6	2
month	7	8	2
year	9	10	2
sectionList	11	32	2
sectionListTuple	12	31	3
sectionName	13	14	4
articles	15	30	4
articlesTuple	16	29	5
toArticle	17	20	6
title	18	19	7
initPage	21	22	6
endPage	23	24	6
authors	25	28	6
author	26	27	7
ProceedingsPage	34	79	1
volume	35	36	2
number	37	38	2
month	39	40	2
year	41	42	2
conference	43	44	2
date	45	46	2
confyear	47	48	2
location	49	50	2
sectionList	51	78	2
sectionListTuple	52	77	3
sectionName	53	54	4
articles	55	76	4
articlesTuple	56	75	5
title	57	58	6
authors	59	62	6
author	60	61	7
initPage	63	64	6
endPage	65	66	6
Toindex	67	70	6
index	68	69	7
fullText	71	74	6
size	72	73	7

Index table for partition 3

Tag	Start	End	Level
virt_root	1	24	0
SigmodRecord	2	23	1
issue	3	22	2
volume	4	5	3
number	6	7	3
articles	8	21	3
article	9	20	4
title	10	11	5
initPage	12	13	5
endPage	14	15	5
authors	16	19	5
author	17	18	6

Index table for partition 4

Appendix III

Pseudo code for the query manager:

```
// Initialization
init() {
    Load the index table as Hash table <tagname, triplet index>
}

// Query Manager
QueryManager(XQuery expression) {
    initializes the list_of_for_clause with ID = variable;
    while (not finished) {
        retrieve whole line from expression -> strline;
        switch (strline) {
            case 'for-clause':
                identify variables involved;
                add strline to list_of_for_clause with ID = variable involved;
                break;
            case 'where-clause':
                parse(strline);
                break;
        }
    }
    /* Query rewriting */
    for each variable {
        build 'for-clause' statement based on bitset_partition result
        replace original 'for-clause' by revised 'for-clause' statement
    }
    return revised_XQuery_expression;
}
```

```
// Parse the where-clause statement
parse(where-clause statement) {
    initializes bitset_partition for each variables;
    extracts XPath and stores them in List_of_XPath;
    for each XPath in List_of_XPath {
        determines the variable associated;
        extracts the tag name into Queue_tag;
        extracts the structural relationship into Queue_struct;
        /* Validate with each partition */
        for each index table(partition) p{
            left_tag = dequeue(Queue_tag);
            right_tag = dequeue(Queue_tag);
            if (not exist(left_tag)) {
                p is not included;
                continue; /* Skip the test for current partition p */
            }
            while (not isEmpty(Queue_tag)) {
                if (not exist(right_tag)) {
                    p is not included;
                    continue; /* Skip the test for current partition p */
                }
                /* Validate structural relationship */
                Relationship = dequeue(Queue_struct);
                switch (relationship) {
                    case Ancestor / Descendant
                        if (not isAncestor_Descendant(left_tag, right_tag)) {
                            p is not included;
                            continue; /* Skip the test for current partition p */
                        }
                        break;
                    case Ancestor / Descendant
                        if (not isParent_Child(left_tag, right_tag)) {
                            p is not included;
                            continue; /* Skip the test for current partition p */
                        }
                        break;
                }
                left_tag = right_tag;
                right_tag = dequeue(Queue_tag);
            } // End of while-loop
        } // End of for-loop (each partition)
        Update bitset_partition for corresponding variable
    } // End of for-loop (each XPath expression)
}

isAncestor_Descendant(left_tag, right_tag) {
    retrieve list_of_left_indices using key = left_tag;
    retreeve list_of_right_indices using key = right_tag;
    for each index from list_of_left_indices (left_index){
        for each index from right_of_left_indices (right_index){
            if (left_index.start ≤ right_index.start ≤ left_index.end) {
                return true;
            }
        }
    }
    Return false;
}
```

```
isParent_Child(left_tag, right_tag) {
    retrieve list_of_left_indices using key = left_tag;
    retreeve list_of_right_indices using key = right_tag;
    for each index from list_of_left_indices (left_index){
        for each index from right_of_left_indices (right_index){
            if (left_index.start ≤ right_index.start ≤ left_index.end
                and left_index.level = right_index.level - 1) {
                return true;
            }
        }
    }
    Return false;
}
```