Abstract

During the last decade, data mining, also known as knowledge discovery in databases (KDD), has been becoming one of the dominant research trends in the fields of computer science and knowledge engineering. Many research results have been proposed and successfully applied to the real world demonstrates that data mining is an active research area based on a solid background rather than an impulsive and volatile one as some people once suspected.

Data mining includes several approaches. Almost all techniques used in this area are borrowed from database, machine learning, artificial intelligence, information theory, probability & statistics, and high performance computing (including parallel computing). There is a wide range of problems need to be solved in this area such as classification/prediction, clustering, association rules mining, sequence mining. Data mining is also an interdisciplinary field because its research results are put into practice in commerce, finance & stock market, biology, medical treatment, education, telecommunication, and so on.

Being aware of the potential of this research domain, I make my choice of "Parallel Mining of Fuzzy Association Rules" for my master thesis. This dissertation not only summarizes the previous work but also remarks the delicate relationship between fuzzy association rules and fuzzy logic theory. In addition, the dissertation proposes a novel parallel algorithm for fuzzy association rules mining.

Roadmap: the thesis is organized as follows:

Chapter one presents an overview of data mining. It encompasses both several descriptive definitions of data mining and main steps in KDD. Also, This section mentions major approaches and techniques as well as classification of data mining systems according to different criteria. The conclusion of this chapter enumerates focused issues during the last few years (probably in next time) and outlines the chief applications of this research domain.

Chapter two formally describes the problem of *association rules mining* by giving basic concepts such as frequent itemset, minimum support, minimum confidence,

etc. Furthermore, this section sums up significant proposals in this field within its 10-year history. A complete understanding of this chapter means the readers will easily perceive next two chapters.

Chapter three presents *fuzzy association rules*. We first state the issue of mining quantitative and categorical association rules together with corresponding methods of data discretization. Quantitative association rules, however, exposes some drawbacks such as "sharp boundary problem" and non-intuitive interpretation. Fuzzy association rule was recommended to overcome these disadvantages. Besides the summarization of the previous research about this kind of association rule, this chapter tries to highlight the subtle relevance between fuzzy association rule and fuzzy theory by determining which functions are suited for T-norm operator. Finally, this section suggests a method to convert fuzzy association rules into quantitative ones based on a threshold w_f of the membership function associated with the fuzzy set of each fuzzy attribute.

Chapter four focuses on *parallel mining of association rules*. The first part of this chapter sums up previously proposed parallel algorithms that were successfully experimented. Most of these algorithms have common shortcomings such that they must strongly perform either data communication or synchronization among related processors during the period of mining frequent itemsets. Making the most of distinct features of fuzzy association rules, I suggest a new parallel algorithm for mining this kind of association rules that significantly decrease the amount of data or control information need to be communicated or synchronized. In addition, this algorithm pays attention to load-balancing among processors thanks to an appropriate strategy in dividing the set of candidate itemsets.

Chapter five concludes the dissertation by summing up the achievements throughout the previous four chapters. Furthermore, this chapter reveals several complex issues that solutions to them remain unconvincing. Finally, I will outline the future research topics.

Acknowledgements

My deepest thank must first go to my research advisor, Dr Ha Quang Thuy, who offers me an endless inspiration in scientific research, leading me to this research area. I particularly appreciate his unconditional support and advice in both academic environment and daily life during the last four years.

Many thanks to all lecturers teaching me throughout my master program such as Prof. Huynh Huu Tue, Prof, Dr Nguyen Xuan Huy, Assoc Prof, Dr Ngo Quoc Tao, Dr Vu Duc Thi, Dr Nguyen Kim Anh, etc. Also, I would like to thank scientists as well as members of management committee of class $K8T_1$: Academic, Prof. Nguyen Van Hieu, Prof, Dr Bach Hung Khang, Assoc Prof, Dr Ho Sy Dam, Prof, Dr Pham Tran Nhu, and Assoc Prof, Dr Do Duc Giao.

My thanks also go to all member of seminar group "*data mining & parallel computing*" such as Dr Do Van Thanh, Msc Pham Tho Hoan, Msc Doan Son, Bsc Bui Quang Minh, Msc Nguyen Tri Thanh, Bsc Nguyen Thanh Trung, Bsc Tao Thi Thu Phuong, Bsc Vu Boi Hang, etc. They are either my teachers or my friends sharing common research topics. Again, I would like to thank them for their technical advice and spiritual supports.

I highly acknowledge the invaluable support and advice in both technical and daily life of my teachers, my colleagues in Department of Information Systems, Faculty of Technology, Vietnam National University, Hanoi such as Dr Nguyen Tue, Assoc Prof, Dr Trinh Nhat Tien, Msc Nguyen Quang Vinh, Msc Vu Ba Duy, Msc Le Quang Hieu, etc.

Finally, I would specially like to say thanks to all members in my family, all my friends. They are really an endless encouragement in my life.

Author of master thesis

Phan Xuan Hieu

Table of Contents

Abstract 1

Acknowledgements 3

Table of Contents 4

List of Figures 6

List of Tables 7

Notations & Abbreviations 8

Chapter 1. Introduction to *Data Mining* 9

1.1 Data Mining 9

1.1.1 Motivation: Why Data Mining? 9

1.1.2 Definition: What is Data Mining? 10

1.1.3 Main Steps in Knowledge Discovery in Databases (KDD) 11

1.1 Major Approaches and Techniques in Data Mining 12

1.2.1 Major Approaches and Techniques in Data Mining 12

1.2.2 Kinds of data could be mined? 13

1.2 Application of Data Mining 14

1.2.1 Application of Data Mining 14

1.2.2 Classification of Data Mining Systems 14

1.3 Focused issues in Data Mining 15

Chapter 2. Association Rules 17

2.1 Motivation: Why Association Rules? 17

2.2 Association Rules Mining – Problem Statement 18

2.3 Main Research Trends in Association Rules Mining 20

Chapter 3. Fuzzy Association Rules Mining 23

3.1 Quantitative Association Rules 23

3.1.1 Association Rules with Quantitative and Categorical Attributes 23
3.1.2 Methods of Data Discretization 24
3.2 Fuzzy Association Rules 27
3.2.1 Data Discretization based on Fuzzy Set 27
3.2.2 Fuzzy Association Rules 29
3.2.3 Algorithm for Fuzzy Association Rules Mining 34
3.2.4 Relation between Fuzzy Association Rules and Quantitative Association Rules 39
3.2.5 Experiments and Conclusions 39
Chapter 4. Parallel Mining of Fuzzy Association Rules 41
4.1 Several Previously Proposed Parallel Algorithms 42
4.1.1 Count Distribution Algorithm 42
4.1.2 Data Distribution Algorithm 43
4.1.3 Candidate Distribution Algorithm 45
4.1.4 Algorithm for Parallel Generation of Association Rules 48
4.1.5 Other Parallel Algorithms 50
4.2 A New Parallel Algorithm for Fuzzy Association Rules Mining 50
4.2.1 Our Approach 51
4.2.2 The New Algorithm 55
4.3 Experiments and Conclusions 55
Chapter 5. Conclusions 56
5.1 Achievements throughout the dissertation 56
5.2 Future research 57
Reference 59

List of Figures

Figure 1 - The volume of data strongly increases in the past two decades 9

Figure 2 - Steps in KDD process 12

Figure 3 - Illustration of an association rule 17

Figure 4 - "Sharp boundary problem" in data discretization 26

Figure 5 - Membership functions of "Age_Young", "Age_Middle-aged", and "Age_Old" 27

Figure 6 - Membership functions of "Cholesterol_Low" and "Cholesterol_High" 28

Figure 7 - Count distribution algorithm on a 3-processor parallel system 43

Figure 8 - Data distribution algorithm on a 3-processor parallel system 45

List of Tables

Table 1 - An example of transactional databases 18

Table 2 - Frequent itemsets in sample database in table 1 with support = 50%.... 18

Table 3 - Association rules generated from frequent itemset ACW 19

Table 4 - Diagnostic database of heart disease on 17 patients..... 23

Table 5 - Data discretization for categorical or quantitative attributes having finite values 25

Table 6 - Data discretization for "Serum cholesterol" attribute..... 25

Table 7 - Data discretization for "Age" attribute 25

Table 8 - Diagnostic database of heart disease on 13 patients..... 29

Table 9 - Notations used in fuzzy association rules mining algorithm..... 35

Table 10 - Algorithm for mining fuzzy association rules..... 35

Table 11 - T_F: Values of records at attributes after fuzzifying 36

Table 12 - C_1: set of candidate 1-itemsets 37

Table 13 - F_2: set of frequent 2-itemsets..... 38

Table 14 - Fuzzy association rules generated from database in table 8..... 39

Table 15 - The sequential algorithm for generating association rules..... 49

Table 16 - Set of fuzzy attributes received after being fuzzified the database in table 8 51

Table 17 - Fuzzy attributes dividing algorithm among processors..... 54

Notations & Abbreviations

Word or phrase	Abbreviation
Knowledge Discovery in Databases	KDD

Chapter 1. Introduction to *Data Mining*

1.1 Data Mining

1.1.1 Motivation: Why Data Mining?

The past two decades has seen a dramatic increase in the amount of information or data being stored in electronic devices (i.e. hard disk, CD-ROM, magnetic tape, etc.). This accumulation of data has taken place at an explosive rate. It has been estimated that the amount of information in the world doubles every two years and the size and number of databases are increasing even faster. Figure 1 illustrates the data explosion [3].

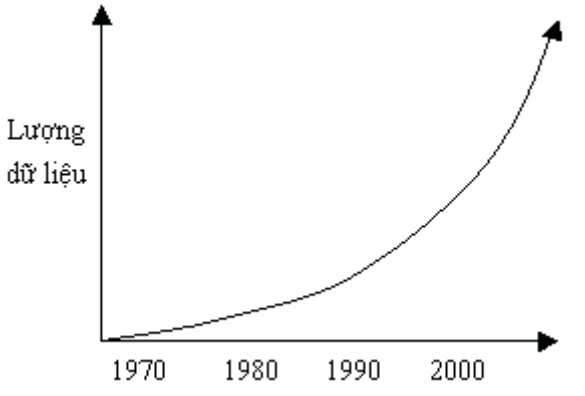

Figure 1 - The volume of data strongly increases in the past two decades

We are drowning in data, but starving for useful knowledge. The vast amount of accumulated data is actually a valuable resource because information is the vital factor for business operations, and decision-makers could make the most of the data to gain precious insight into the business before making decisions. Data mining, the extraction of hidden predictive information from large databases, is a powerful new technology with great potential to help companies focus on the most significant information in their data collection (databases, data warehouses, data repositories). The automated, prospective analyses offered by data mining go beyond the normal analyses of past events provided by retrospective tools typical of decision support systems. Data mining tools can answer business questions that traditionally were too time-consuming to resolve. This is where data mining & knowledge discovery in databases demonstrates its obvious benefits for today's

competitive business environment. Nowadays, Data Mining & KDD has been becoming a key role in computer science and knowledge engineering areas.

The initial application of data mining is only in commerce (retail) and finance (stock market). However, data mining is now widespreadly and successfully put into other fields such as bio-informatics, medical treatment, telecommunication, education, etc.

1.1.2 Definition: What is Data Mining?

Before listing some definitions of data mining, I would like to present a small explanation about terminology so that readers could avoid unnecessary confusions. As mentioned above, we can roughly understand that data mining is a process of extracting non-trivial, implicit, previously unknown, and potentially useful knowledge from huge sets of data. Thus, we should name this process is knowledge discovery in database (KDD) instead of data mining. However, most of the researchers agree that the two above terminologies (data mining and KDD) are similar and they can be used interchangeably. They explain for this "humorous" misnomer that the core motivation of KDD is useful knowledge, but the main object they have to deal with is data. Therefore, in a sense, data mining and KDD have the same meaning.

However, data mining is sometimes refered to as one step in the whole KDD process [3] [43].

There are numerous definitions of data mining and they are all descriptive. I would like to restate herein some of them that are widely accepted.

Definition one: William J Frawley, Gregory Piatetsky-Shapiro, and Christopher J Matheus 1991 [43]:

> *"Knowledge discovery in databases, also known Data mining, is the non-trivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data."*

Definition two: Marcel Holshemier và Arno Siebes (1994):

> *"Data Mining is the search for relationships and global patterns that exist in large databases but are 'hidden' among the vast amount of data, such as a*

relationship between patient data and their medical diagnosis. These relationships represent valuable knowledge about the database and the objects in the database and, if the database is a faithful mirror, of the real world registered by the database."

1.1.3 Main Steps in Knowledge Discovery in Databases (KDD)

The whole KDD process is usually decomposed into the following steps [3] [14] [23]:

- Data selection: selecting or segmenting the necessary data that needs to be mined from large data sets (databases, data warehouses, data repositories) according to some criteria.
- Data preprocessing: this is the data clean and reconfiguration stage where some techniques are applied to deal with incomplete, noisy, and inconsistent data. This step also tries to reduce data by using aggregate and group function, data compression methods, histograms, sampling, etc. Furthermore, discretization techniques (bining, histograms, cluster analysis, entropy-based discretization, segmentation) can be used to reduce the number of values for a given continuous attribute by dividing the range of the attribute into separated intervals. After this step, data is clean, complete, uniform, reduced, and discretized.
- Data transformation: in this step, data are transformed or consolidated into forms appropriate for mining. Data transformation can involve data smoothing and normalization. After this step, data are ready for the mining step.
- Data mining: this is considered to be the most important step in KDD process. It applies some data mining techniques (chiefly borrowing from machine learning and other fields) to discover and extract useful patterns or relationships from data.
- Knowledge representation and evaluation: the patterns identified by the system in previous step are interpreted into knowledge which can then be used to support human decision-making (e.g. prediction and classification tasks, summarizing the contents of a database or explaining observed phenomena).

Knowledge representation also converts patterns into user-readable expressions such as trees, graphs, charts & tables, rules, etc.

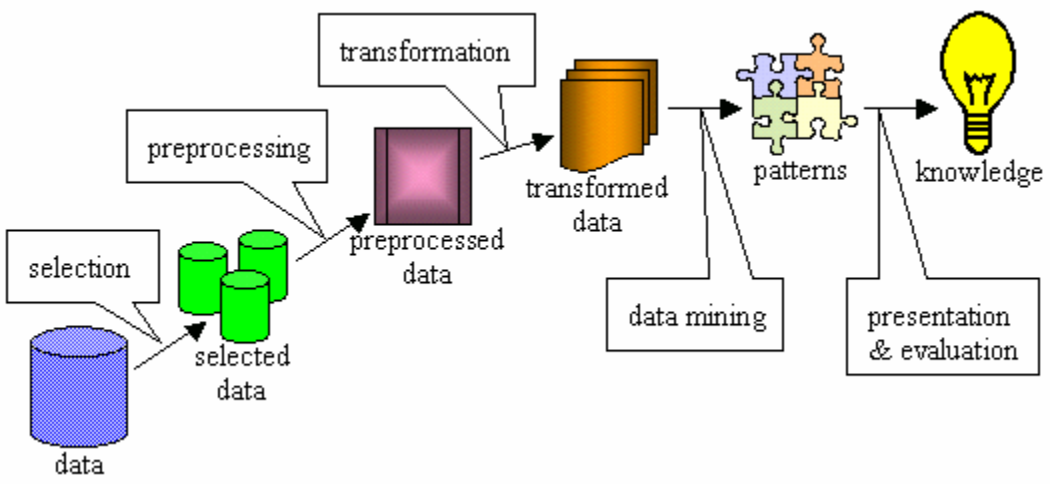

Figure 2 - Steps in KDD process

1.1 Major Approaches and Techniques in Data Mining

1.2.1 Major Approaches and Techniques in Data Mining

Data mining consists of many approaches. They can be classified according to functionality, kind of knowledge, type of data to be mined, or whatever appropriate criteria [14]. I describe major approaches bellow:

- Classification & prediction: this method tries to arrange a given object into an appropriate class among the others. The number of classes and their name are definitely known. For example, we can classify or anticipate geographic regions according to weather and climate data. This approach normally uses typical techniques and concepts in machine learning such as decision tree, artificial neural network, k-min, support vector machine, etc. Classification is also called supervised learning.
- Association rules: this is a relatively simple form of rule e.g. “80 percent of men that purchase beer also purchase dry-beef”. Association rule is now

successfully applied in supermarket (retail), medicine, bio-informatics, finance & stock market, etc.

- Sequential/temporal patterns mining: this method is somewhat similar to association rules except that data and mining results (a kind of rule) always contain a temporal attribute to exhibit the order or sequence in which events or objects effect upon each other. This approach plays a key role in finance and stock market thanks to its capability of prediction.
- Clustering & segmentation: this method tries to arrange a given object into a suited category (also known as cluster). The number of clusters may be dynamic and their labels (names) are unknown. Clustering and segmentation are also called unsupervised learning.
- Concept description & summarization: the main objective of this method is to describe or summarize an object so that the obtained information is compact and condensed. Document or text summarization may be a typical example.

1.2.2 Kinds of data could be mined?

Data mining can work on various kinds of data. The most typical data types are as follows:

- Relational databases (managed by relational database management systems such as MS SQL Server, Oracle, DB2, etc).
- Data warehouses & data repositories (including multidimensional data structures such as data cube, etc.)
- Transactional databases (data are organized as set of transactions, e.g. transactions contain items bought by customers in a supermarket).
- Object - relational databases (a hybrid kind of object-oriented model and relational model).
- Spatial, temporal data, and time-series data.
- Multimedia databases (containing data related to diverse type such as text, image, audio, and video, etc).

- Text and Web databases (documents in various formats such as *.doc, *.pdf, *.ps, *.html, etc).

1.2 Application of Data Mining

1.2.1 Application of Data Mining

Although data mining is a relatively new research trend, it is a big attraction of researchers because of its practical applications in many areas. The following should be typical applications:

- Data analysis and decision-marking supports. This application is popular in commerce (retail industry), finance & stock market, etc.
- Medical treatment: finding the potential relevance among symptoms, diagnoses, and treatment methods (nutrient, prescription, surgeon, etc).
- Text and Web mining: document summarization, text retrieval and text searching, text and hypertext classification.
- Bio-informatics: search and compare typical or special genetic information such as genomes and DNA, the implicit relations between a set of genomes and a genetic disease, etc.
- Finance & stock market: examining data to extract predicted information for price of a certain kind of coupon.
- Others (telecommunication, medical insurance, astronomy, anti-terrorism, sports, etc).

1.2.2 Classification of Data Mining Systems

Data mining is a knowledge engineering related field that involves many others research areas such as database, machine learning, artificial intelligence, high performance computing, data & knowledge visualization, etc. We could classify data mining systems according to different criteria as follows:

- Classifying based on kind of data to be mined: data mining systems work with relational databases, data warehouses, transactional databases, object-oriented

databases, spatial and temporal databases, multimedia databases, text and web databases, etc.

- Classifying based on type of mined knowledge: data mining tools that return summarization or description, association rules, classification or prediction, clustering, etc.
- Classifying based on what kind of techniques to be used: data mining tools work as online analytical processing (OLAP) systems, use machine learning techniques (decision tree, artificial neural network, k-min, genetic algorithm, support vector machine, rough set, fuzzy set, etc.), data visualization, etc.
- Classifying based on what fields the data mining systems are applied to: data mining systems are used in different fields such as commerce (retail industry), telecommunication, bio-informatics, medical treatment, finance & stock market, medical insurance, etc.

1.3 Focused issues in Data Mining

Data mining is a relatively new research topic. Thus, there are several pending or unconvincingly solved issues. I relate herein some of them that are attracting attention of data mining researchers.

- OLAM (Online Analytical Mining) – a smooth combination of databases, data warehouses, and data mining. Nowadays, database management systems like Oracle, MS SQL Server, IBM DB2 have integrated OLAP and data warehouse functionalities to facilitate users in data retrieval and data analyzing. These add-in supports also charge users an additional sum of money. Researchers in these fields hope to go beyond the current limitation by developing multi-purposes OLAM systems that support data transactions for daily business operations as well as data analyzing for making decision [14].
- Data mining systems can mine various forms of knowledge from different types of data [14] [7].
- How to enhance the performance, accuracy, scalability, and integration of data mining systems? How to decrease the computational complexity? How to

improve the ability of dealing with incomplete, inconsistent, and noisy data? Three questions above should still be concentrated in the future [14].

- Taking advantage of background knowledge or knowledge from users (experts or specialists) to upgrade the total performance of data mining systems [7] [1].
- Parallel and distributed data mining is an interesting research trend because it make use of powerful computing systems to reduce response time. This is essential because more and more real-time applications are needed in today's competitive world [5] [8] [12] [18] [26] [31] [32] [34] [42].
- Data Mining Query Language (DMQL): Researchers in this area try to design a standard query language for data mining. This language will be used in OLAM systems as if SQL are widely used in relational databases [14].
- Knowledge representation and visualization are also taken into consideration to express knowledge in human-readable and easy-to-use forms. Knowledge can be represented in more intuitive expressions due to multidimensional or multilevel data structures.

Chapter 2. Association Rules

2.1 Motivation: Why Association Rules?

Association rule is the form of "70 percent of customers that *purchase beer* also *purchase dry beef*, 20 percent of customers purchase both" or "75 percent of patients who *smoke cigarettes* and *live near polluted areas* also *get lung cancer*, 25 percent of patients smoke and live near polluted areas as well as suffer from lung cancer". "purchase beer" or "smoke cigarettes and live near polluted areas" are antecedents, "purchase dry beef" and "get lung cancer" are called consequents of association rules. 20% and 30% are called support factors (percentage of transactions or records that contain both antecedent and consequent of a rule), 70% and 75% are called confidence factors (percentage of transactions or records that hold the antecedent also hold the consequent of a rule). The following figure pictorially depicts the former example of association rules.

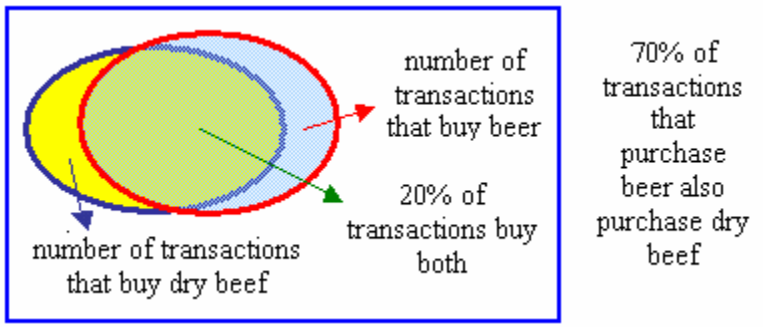

Figure 3 - Illustration of an association rule

The knowledge and information returning from association rules have an obvious difference in meaning from that of normal queries (usual in SQL syntax). This knowledge contains previously unknown relationships and predictions hidden in massive volumes of data. It not only results from usual group, aggregate or sort operations but also results from a complicated and time-consuming computing process.

Being a simple kind of rule, association rules, however, carry useful knowledge and contribute substantially to decision-making process. Unearthing significant rules from databases is the main motivation of researchers.

2.2 Association Rules Mining – Problem Statement

Let $\mathbf{I} = \{i_1, i_2, \ldots, i_n\}$ be set of *n* items or attributes (in transactional or relational databases respectively) and $\mathbf{T} = \{t_1, t_2, \ldots, t_m\}$ be set of *m* transactions or records (in transactional or relational databases respectively). Each transaction is identified with its unique TID number. A (transactional) database **D** is a binary relation δ on the Descart multiplication **I**x**T** (or also written $\delta \subseteq \mathbf{I}x\mathbf{T}$). If an item **i** occurs in a transaction **t**, we write $(\mathbf{i}, \mathbf{t}) \in \delta$ or $\mathbf{i}\delta\mathbf{t}$. Generally speaking, a transactional database is a set of transactions, where each transaction **t** contains a set of items or $\mathbf{t} \in 2^{\mathbf{I}}$ (where $2^{\mathbf{I}}$ is power set of **I**) [24] [36].

For example, consider the sample transactional database shown in table 1 with **I** = {A, C, D, T, W} and **T** = {1, 2, 3, 4, 5, 6}.

Transaction ID	Itemset
1	A C T W
2	C D W
3	A C T W
4	A C D W
5	A C D T W
6	C D T

Table 1 - An example of transactional databases

$\mathbf{X} \subseteq \mathbf{I}$ is called an itemset. The *support factor* of an itemset **X**, denoted as *s*(**X**), is the percentage of transactions that contains **X**. An itemset **X** is *frequent* if its support is greater than or equal to a user-specified *minimum support* (*minsup*) value, i.e. $s(\mathbf{X}) \geq minsup$ [36].

The following table enumerates all possible frequent itemsets in sample database in table 1 with *minsup* value is 50%.

Frequent itemsets	Support factor
C	100% (6)
W, CW	83% (5)
A, D, T, AC, AW, CD, CT, ACW	67% (4)
AT, DW, TW, ACT, ATW, CDW, CTW, ACTW	50% (3)

Table 2 - Frequent itemsets in sample database in table 1 with support = 50%

Association rule is an implication in the form of $X \xrightarrow{c} Y$, where **X** and **Y** are frequent itemsets that disjoint, i.e. $\mathbf{X} \cap \mathbf{Y} = \varnothing$, and *c*, confidence factor of the rule, is the conditional probability that a transaction contains **Y**, given that it contains **X**, i.e. $c = s(\mathbf{X} \cup \mathbf{Y}) / s(\mathbf{X})$. A rule is *confident* if its confidence factor is larger or equal to a user-specified *minimum confidence* (*minconf*) value, i.e. $c \geq minconf$ [36].

The association rules mining task can be stated as follows:

> Let **D** be a (transactional) database, *minsup* and *minconf* are minimum support and minimum confidence respectively. The mining task try to discover all *frequent* and *confident* association rules $X \rightarrow Y$, i.e. $s(\mathbf{X} \cup \mathbf{Y}) \geq minsup$ and $c(X \rightarrow Y) = s(\mathbf{X} \cup \mathbf{Y}) / s(\mathbf{X}) \geq minconf$.

Almost all previously proposed algorithms decompose this mining task into two separated phases [4] [5] [20] [24] [34] [35]:

- Phase one: finding all possible frequent itemsets from database **D**. This stage is usually complicated and time-consuming because it needs much time of CPU (CPU-bound) and I/O operations (I/O-bound).
- Phase two: generating confident association rules from discovered frequent itemsets in the previous step. If **X** is a frequent itemset, confident association rules created from **X** have the form of $X' \xrightarrow{c} X \setminus X'$, where **X'** is any non-empty subset of **X** and **X** \ **X'** is subtraction of **X'** from **X**. This step is relatively straightforward and much less time-consuming than the step one.

The following table lists all possible association rules generated from the frequent itemset ACW (from database in table 1) with *minconf* = 70%.

Association rules	Satisfy *minconf* ≥ 70%?
$A \xrightarrow{100\%} CW$	Yes
$C \xrightarrow{67\%} AW$	No
$W \xrightarrow{80\%} AC$	Yes
$AC \xrightarrow{100\%} W$	Yes
$AW \xrightarrow{100\%} C$	Yes
$C \xrightarrow{80\%} AW$	Yes

Table 3 - Association rules generated from frequent itemset ACW

2.3 Main Research Trends in Association Rules Mining

Since proposed by R. Agrawal in 1993 [36], the field of association rules mining has developed into various directions thanks to a variety of improvements from researchers. Some of proposals are try to improve the precision and performance, some try to tune the interestingness of rules, etc. I list herein some of dominant trends.

- Mining binary or boolean association rules: this is the initial research direction of association rules. Most of the early mining algorithms are related to this kind of rule [20] [35] [36]. In binary association rules, an item is only determined whether it is present or not. The quantity associated with each item is totally ignored, e.g. a transaction buying twenty bottles of beer is the same a transaction that buys only one bottle. The most typical algorithms mining binary association rules are Apriori together with its variants (Apriori-Tid and AprioriHybrid) [35]. An example of this type of rule is "*bying bread = 'yes'* AND *buying sugar = 'yes' => buying milk = 'yes'* AND *buying butter = 'yes'*, with support 20% and confidence 80%"
- Quantitative and categorical association rules: attributes in databases may be binary (boolean), number (quantitative), or nominal (categorical), etc. To discover association rules that involve these data types, quantitative and categorical attributes need to be discretized to convert into binary ones. There exist some of discretization methods that are proposed in [34] [39]. An example of this kind of rule is "*sex = 'male'* AND *age ∈ '50..65'* AND *weight ∈ '60..80'* AND *sugar in blood > 120mg/ml => blood pressure = 'high'*, with support 30% and confidence 65%".
- Fuzzy association rules: this type of rule was proposed to overcome several drawbacks in quantitative association rules such as "sharp boundary problem" or semantic expression. Fuzzy association rule is more natural and intuitive to users thanks to its "fuzzy" characteristics. An example is "*dry cough = 'yes'* AND ***high fever*** AND *muscle aches = 'yes'* AND *breathing difficulties = 'yes'* => *get SARS* (Severe Acute Respiratory Syndrome) = 'yes', with support 4%

and confidence 80%". ***High fever*** in the above rule is a fuzzy attribute. We measure the body temperature based on a fuzzy concept.

- Multi-level association rules: all kinds of association rules above are too concrete, so they cannot reflect relationships on general views. Multi-level or generalized association rule is devised to surmount this problem [15] [37]. In approach, we would prefer rule like "*buy PC = 'yes' => buy operating system = 'yes'* AND *buy office tools = 'yes'*" rather than "*buy IBM PC = 'yes' => buy Microsoft Windows = 'yes'* AND *buy Microsoft Office = 'yes'*". Obviously, the former rule is the generalized form of the latter and the latter is the specific form of the former.

- Association rules with weighted items (or attributes): we use weight associated with each item to indicate "the level" that item contributes to the rule. In other words, weights are used to measure the importance of items. For example, while surveying SARS plague within a certain group of people, the information of *body temperature* and *respiratory system* is much more essential than that of *age*. To reflect the difference between the above attributes, we attach greater weight values for *body temperature* and *respiratory system* attributes. This is an attractive research direction and solutions to it were presented in several papers [10] [44]. By using weights, we should discover scarce association rules of high interestingness. This means that we can retain rules with small supports but have a special meaning.

- Besides examining of variants of association rules, researchers pay attention to how to accelerate the phase of finding frequent itemsets. Most of recommended algorithms are try to reduce the number of frequent itemsets need to be mined by developing new theories of *maximal frequent itemsets* [11] (MAFIA algorithm), *closed itemsets* [13] (CLOSET algorithm), [24] (CHARM algorithm), [30]. These new approaches remarkably decrease mining time owning to their "delicate pruning strategies". Experi-ments show that these algorithms outperform known ones like Apriori, AprioriTid, etc.

- Parallel algorithms for association rules mining: in addition to sequential or serial algorithms, parallel algorithms are invented to enhance total performance of mining process by making use of robust parallel systems. The appearance of

parallel and distributed data mining is really reasonable because size of databases increases sharply and real-time applications are widely used in recent years. Numerous parallel algorithms for mining association rules were devised during ten past years in [5] [12] [18] [26] [31] [32] [34]. They are both platform dependent and platform independent.

- Mining association rules in the point of view of rough set theory [41].
- Furthermore, there exist other research trends such as online association rule mining [33] that data mining tools are integrated or directly connected to data warehouses or data repositories based on well-known technologies as OLAP, MOLAP, ROLAP, ADO, etc.

Chapter 3. Fuzzy Association Rules Mining

3.1 Quantitative Association Rules

3.1.1 Association Rules with Quantitative and Categorical Attributes

Mining quantitative and categorical association rules is an important task because of its practical applications on real world databases. This kind of association rules first introduced in [38].

The following table demonstrates a shortened real database about diagnosis of heart disease. Its attributes are represented in different formats including boolean (binary), number (quantitative), or nominal (categorical).

Age	Sex	Chest pain type (1, 2, 3, 4)	Serum cholesterol (mg/ml)	Fasting blood sugar (>120mg/ml)	Resting electrocardio-graphics (0, 1, 2)	Maximum heart rate	Heart disease
60	1 (f)	4	206	0(<120mg/ml)	2	132	2 (yes)
54	1	4	239	0	0	126	2
54	1	4	286	0	2	116	2
52	1	4	255	0	0	161	2
68	1	3	274	1(>120mg/ml)	2	150	2
54	1	3	273	0	2	152	1 (no)
54	0 (m)	2	288	1	2	159	1
67	0	3	277	0	0	172	1
46	0	2	204	0	0	172	1
52	1	2	201	0	0	158	1
40	1	4	167	0	2	114	2
37	1	3	250	0	0	187	1
71	0	2	320	0	0	162	1
74	0	2	269	0	2	121	1
29	1	2	204	0	2	202	1
70	1	4	322	0	2	109	2
67	0	3	544	0	2	160	1

Table 4 - Diagnostic database of heart disease on 17 patients

In the above database, three attributes *Age, Serum cholesterol*, *Maximum heart rate* are quantitative, two attributes *Chest pain type* and *resting electrocardio-graphics* are categorical, and all the rest are binary (*Sex*, *Heart disease*, *Fasting blood sugar*). In fact, binary data type is also considered to be a special form of category. From the data in table 4, we can extract such rules as:

<Age: 54..74> AND **<Sex: Female>** AND **<Cholesterol: 200..300>** => **<Heart disease: Yes>**, with support 23.53% and confidence 80%.

<Sex: Male> AND **<Resting electrocardiographics: 0>** AND **<Fasting blood sugar ≤ 120>** => **<Heart disease: No>**, with support 17.65% and confidence 100%.

The approach proposed in [34] discovers this kind of rules by partitioning value ranges of quantitative and categorical attributes into separated intervals to convert them into binary ones. Traditional well-known algorithms such as Apriori [35], CHARM [24], COLSET [20] can then work on these new binary attri-butes as original problem of mining boolean association rules.

3.1.2 Methods of Data Discretization

Binary association rules mining algorithms [20] [24] [35] [36] only work with relational database containing only binary attributes or transac-tional databases as shown in table 1. They cannot be applied directly to practical databases as shown in table 4. To conquer this obstacle, quantitative and categori-cal columns must first be converted into boolean ones [34] [39]. However, there are some limitations in data discretization that influence the quality of discovered rules. The output rules do not satisfy researchers's expectation. The following section describes major discretization methods to contrast their dis-advantages.

- Let A be a discrete quantitative or categorical attribute with finite value domain $\{v_1, v_2, \ldots, v_k\}$ and k is small enough ($k < 100$). After being discretized, the original attribute is developed into k new binary attributes named A_V_1, A_V_2, … A_V_k. Value of a record at column A_V_i is equal to *True* (Yes or 1) if the original value of this record at attribute A is equal to v_i, all the rest cases will set the value of A_V_i to *False* (No or 0). The attributes *Chest pain type* and *resting electrocardiographics* in table 4 belong to this case. After transforming, the initial attribute *Chest pain type* will be converted into four binary columns *Chest_pain_type_1*, *Chest_pain_type_2*, *Chest_pain-_type_3*, *Chest_pain_type_4* as shown in the following table.

Chest pain type (1, 2, 3, 4)	➔	Chest_pain_ type_one_1	Chest_pain_ type_one_2	Chest_pain_ type_one_3	Chest_pain_ type_one_4
4		0	0	0	1
1	sau khi	1	0	0	0
3	rời rạc	0	0	1	0
2	hóa	0	1	0	0

Table 5 - Data discretization for categorical or quantitative attributes having finite values

- If A is a continuous and quantitative attribute or a categorical one having value domain $\{v_1, v_2, \ldots, v_p\}$ (p is relatively large). A will be mapped to q new binary columns in the form of <A: $start_1..end_1$>, <A: $start_2..end_2$>, …, <A: $start_q..end_q$>. Value of a given record at column <A: $start_i..end_i$> is *True* (Yes or 1) if the original value v at this record of A is between $start_i$ and end_i, <A: $start_i..end_i$> will receive *False* (No or 0) value for all other cases of v. The attributes *Age*, *Serum cholesterol*, and *Maximum heart rate* in table 4 belong to this form. *Serum cholesterol* and *Age* could be discretized as shown in the two following tables:

Serum cholesterol	➔	<Cholesterol: 150..249>	<Cholesterol: 250..349>	<Cholesterol: 350..449>	<Cholesterol: 450..549>
544		0	0	0	1
206		1	0	0	0
286		0	1	0	0
322		0	1	0	0

Table 6 - Data discretization for "Serum cholesterol" attribute

Age	➔	<Age: 1..29>	<Age: 30..59>	<Age: 60..120>
74		0	0	1
29		1	0	0
30		0	1	0
59		0	1	0
60		0	0	1

Table 7 - Data discretization for "Age" attribute

Unfortunately, the mentioned discretization methods encounter some pitfalls such as “sharp boundary problem” [4] [9]. The figure below displays the support distribution of an attribute A having a value range from 1 to 10. Supposing that we divide A into two separated intervals [1..5] and [6..10] respectively. If the *minsup* value is 41%, the range [6..10] will not gain sufficient support. Therefore [6..10] can not satisfy *minsup* (40% < *minsup* = 41%) even though there is a large support near its left boundary. For example, [4..7] has support 55%, [5..8] has support

45%. So, this partition results in a “sharp boundary” between 5 and 6, and therefore mining algorithms cannot generate confident rules involving interval [6..10].

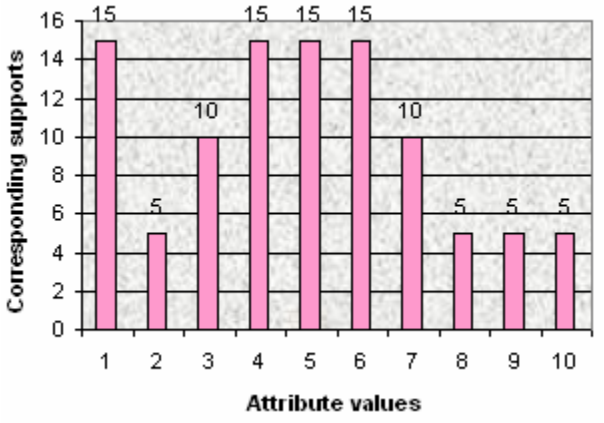

Figure 4 - "Sharp boundary problem" in data discretization

Another attribute partitioning method [38] is to divide the attribute domain into overlapped regions. We can see that the boundaries of intervals are overlapped with each other. As a result, the elements located near the boundary will contribute to more than one interval such that some intervals may become interesting in this case. It is, however, not reasonable because total support of all intervals exceeds 100% and we unintentionally overemphasize the importance of values located near boundaries. This is not natural and inconsistent.

Furthermore, partitioning attribute domain into separated ranges results in a problem in rule interpretation. The table 7 shows that two values 29 and 30 belong to different intervals though they are very similar in indicating old level. Also, supposing that the range [1..29] denotes young people, [30..59] for middle-aged people, and [60..120] for old ones, so the age of 59 implies a middle-aged person whereas the age of 60 implies an old person. This is not intuitive and natural in understanding the meaning of quantitative association rules.

Fuzzy association rule was recommended to overcome the above shortcomings [4] [9]. This kind of rule not only successfully improves “sharp boundary problem” but also allows us to express association rules in a more intuitive and friendly format.

For example, the quantitative rule "**<Age: 54..74>** AND **<Sex: Female>** AND **<Cholesterol: 200..300>** => **<Heart disease: Yes>**" is now replaced by "<**Age_Old**> AND < **Sex: Female**> AND <**Cholesterol_High** > => < **Heart disease: Yes**>". *Age_Old* and *Cholesterol_High* in the above rule are fuzzy attributes.

3.2 Fuzzy Association Rules

3.2.1 Data Discretization based on Fuzzy Set

In the fuzzy set theory [21] [47], an element can belongs to a set with a membership value in [0, 1]. This value is assigned by the membership function associated with each fuzzy set. For attribute x and its domain D_x (also known as universal set), the mapping of the membership function m_{f_x} associated with fuzzy set f_x is as follow:

$$m_{f_x}(x) : D_x \rightarrow [0,1] \quad (3.1)$$

The fuzzy set provides a smooth change over the boundaries and allows us to express association rules in a more expressive form. Let's use the fuzzy set in data discretizing to make the most of its benefits.

For the attribute *Age* and its universal domain [0, 120], we attach with it three fuzzy sets *Age_Young*, *Age_Middle-aged*, and *Age_Old*. The graphic representations of these fuzzy sets are shown in the following figures.

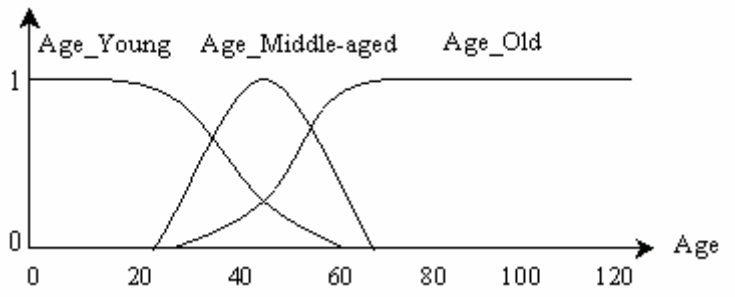

Figure 5 - Membership functions of "Age_Young", "Age_Middle-aged", and "Age_Old"

By using fuzzy set, we completely get rid of "sharp boundary problem" thanks to its own characteristics. For example, the graph in figure 5 indicates that the ages of 59 and 60 have membership values of fuzzy set *Age_Old* approximately 0.85 and

0.90 respectively. Similarly, the ages of 30 and 29 towards the fuzzy set *Age_Young* are 0.70 and 0.75. Obviously, this transformation method is much more intuitive and natural than known discretization ones.

Another example, the original attribute *Serum cholesterol* is decomposed into two new fuzzy attributes *Cholestero_Low* and *Cholestero_High*. The following figure portrays membership functions of these fuzzy concepts.

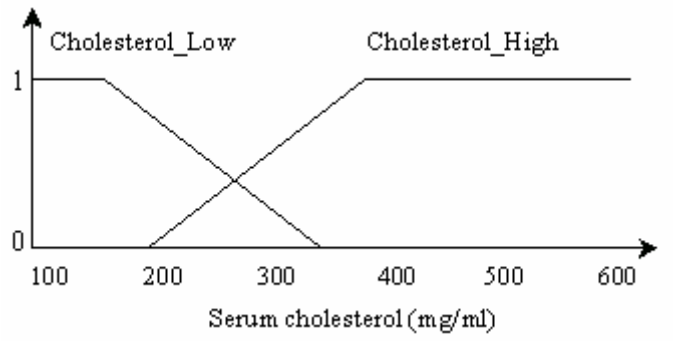

Figure 6 - Membership functions of "Cholesterol_Low" and "Cholesterol_High"

If A is a categorical attribute having value domain $\{v_1, v_2, \ldots, v_k\}$ and k is relatively small, we fuzzify this attribute by attaching a new fuzzy attribute A_V_i to each value v_i. The value of membership function $m_{A\_Vi}(x)$ equals to 1 if $x = v_i$ and equals to 0 for vice versa. Ultimately thinking, A_V_i is also a normal set because its membership function value is either 0 or 1. If k is too large, we can fuzzify this attribute by dividing its domain into intervals and attaching a new fuzzy attribute to each partition. However, developers or users should consult experts for necessary knowledge related to current data to achieve an appropriate division.

Data discretization using fuzzy sets could bring us the following benefits:

- Firstly, smooth transition of membership functions should help us eliminate the "sharp boundary problem".
- Data discretization by using fuzzy sets assists us significantly reduce number of new attributes because number of fuzzy sets associated with each original attribute is relatively small comparing to that of an attribute in quantitative association rules. For instance, if we use normal discretization methods over attribute *Serum cholesterol*, we will obtain five sub-ranges (also five new

attributes) from its original domain [100, 600], whereas we will create only two new attributes *Cholesterol_Low* and *Cholesterol_High* by applying fuzzy sets. This advantage is very essential because it allows us to compact the set of candidate itemsets, and therefore shortening the total mining time.

- Fuzzy association rule is more intuitive, and natural than known ones.
- All values of records at new attributes after fuzzifying are in [0, 1]. This is to imply the possibility that a given element belongs to a fuzzy set. As a result, this flexible coding brings us an exact method to measure the contribution or impact of each record to overall support of an itemset.
- The next advantage that we will see more clearly in the next section is fuzzified databases still hold "downward closure property" (*all subsets of a frequent itemset are also frequent, and any superset of a non-frequent itemset will be not frequent*) if we have a wise choice for T-norm operator. Thus, conventional algorithms such as Apriori also work well upon fuzzified databases with just slight modifications.
- Another benefit is this data discretization method can be easily applied to both relational and transactional databases.

3.2.2 Fuzzy Association Rules

Age	Serum cholesterol (mg/ml)	Fasting blood sugar (>120mg/ml)	Heart disease
60	206	0 (<120mg/ml)	2 (yes)
54	239	0	2
54	286	0	2
52	255	0	2
68	274	1 (>120mg/ml)	2
54	288	1	1 (no)
46	204	0	1
37	250	0	1
71	320	0	1
74	269	0	1
29	204	0	1
70	322	0	2
67	544	0	1

Table 8 - Diagnostic database of heart disease on 13 patients

Let $\mathbf{I} = \{i_1, i_2, \ldots, i_n\}$ be a set of *n* attributes, denoting i_u is the u^{th} attribute in **I**. And $\mathbf{T} = \{t_1, t_2, \ldots, t_m\}$ is a set of *m* records, and t_v is the v^{th} record in **T**. The value

of record t_v at attribute i_u can be refered to as $t_v[i_u]$. For instance, in the table 8, the value of $t_5[i_2]$ (also the value of t_5[*Serum cholesterol*]) is 274 (mg/ml). Using fuzzification method in the previous section, we associate each attribute i_u with a set of fuzzy sets F_{i_u} as follows:

$$F_{i_u} = \left\{ f_{i_u}^1, f_{i_u}^2, ..., f_{i_u}^k \right\} \quad (3.2)$$

For example, with the database in table 8, we have:

$$F_{i_1} = F_{Age} = \{Age\_Young, Age\_Middle\text{-}aged, Age\_Old\} \text{ (với } k = 3)$$

$$F_{i_2} = F_{Serum\_Cholesterol} = \{Cholesterol\_Low, Cholesterol\_High\} \text{ (với } k = 2)$$

A fuzzy association rule [4] [9] ***is an implication in the form of:***

$$\mathbf{X} \text{ is } \mathbf{A} \Rightarrow \mathbf{Y} \text{ is } \mathbf{B} \quad (3.3)$$

Where:

- $\mathbf{X}, \mathbf{Y} \subseteq \mathbf{I}$ are itemsets. $\mathbf{X} = \{x_1, x_2, \ldots, x_p\}$ ($x_i \neq x_j$ if $i \neq j$) and $\mathbf{Y} = \{y_1, y_2, \ldots, y_q\}$ ($y_i \neq y_j$ if $i \neq j$).
- $\mathbf{A} = \{f_{x1}, f_{x2}, \ldots, f_{xp}\}$, $\mathbf{B} = \{f_{y1}, f_{y2}, \ldots, f_{yq}\}$ are sets of fuzzy sets corresponding to attributes in **X** and **Y**. $f_{xi} \in F_{xi}$ và $f_{yj} \in F_{yj}$.

We can rewrite the fuzzy association rules as two following forms:

$$\mathbf{X}=\{x_1, \ldots, x_p\} \text{ is } \mathbf{A}=\{f_{x1}, \ldots, f_{xp}\} \Rightarrow \mathbf{Y}=\{y_1, \ldots, y_q\} \text{ is } \mathbf{B}=\{f_{y1}, \ldots, f_{yq}\} \quad (3.4)$$

or

$$(x_1 \text{ is } f_{x1}) \otimes \ldots \otimes (x_p \text{ is } f_{xp}) \Rightarrow (y_1 \text{ is } f_{y1}) \otimes \ldots \otimes (y_q \text{ is } f_{yq}) \quad (3.5)$$

(where $\otimes$ is T-norm operator in fuzzy logic theory)

A fuzzy itemset is now defined as a pair <**X**, **A**>, in which **X** (⊆ **I**) is an itemset and **A** is a set of fuzzy sets associated with attributes in **X**.

The support of a fuzzy itemset <**X**, **A**> is denoted *fs*(<**X**, **A**>) and determined by the following formula:

$$fs(<X,A>) = \frac{\sum_{v=1}^{m}\left\{\alpha_{x_1}(t_v[x_1]) \otimes \alpha_{x_2}(t_v[x_2]) \otimes \ldots \otimes \alpha_{x_p}(t_v[x_p])\right\}}{|T|} \quad (3.6)$$

Where:

- **X** = {x_1, …, x_p} and t_v is the v^{th} record in **T**.
- ⊗ is the T-norm operator in fuzzy logic theory. Its role is similar to that of logic operator AND in traditional logic.
- $\alpha_{x_u}(t_v[x_u])$ is calculated as follows:

$$\alpha_{x_u}(t_v[x_u]) = \begin{cases} m_{x_u}(t_v[x_u]) & if \quad m_{x_u}(t_v[x_u]) \geq w_{x_u} \\ 0 & if\ vice\ versa \end{cases} \quad (3.7)$$

  m_{x_u} is the membership function of fuzzy set f_{x_u} associated with x_u, and w_{x_u} is a threshold of membership function m_{x_u} and specified by users

- |**T**| (card of **T**) is the total number of records in **T** (also equal to *m*).

A frequent fuzzy itemset: a fuzzy itemset <**X**, **A**> is frequent if its support is greater or equal to a fuzzy minimum support (*fminsup*) specified by users, i.e.

$$fs(<\mathbf{X}, \mathbf{A}>) \geq fminsup \quad (3.9)$$

The support of a fuzzy association rule is defined as follows:

$$fs(<\mathbf{X} \text{ is } \mathbf{A} => \mathbf{B} \text{ is } \mathbf{Y}>) = fs(<\mathbf{X} \cup \mathbf{Y}, \mathbf{A} \cup \mathbf{B}>) \quad (3.10)$$

A fuzzy association rule is frequent if its support is larger or equal to *fminsup*, i.e. *fs*(<**X** is **A** => **B** is **Y**>) ≥ *fminsup*.

Confidence factor of a fuzzy association rule is denoted *fc*(**X** is **A** => **Y** is **B**) and defined as:

$$fc(\mathbf{X} \text{ is } \mathbf{A} \Rightarrow \mathbf{Y} \text{ is } \mathbf{B}) = fs(<\mathbf{X} \text{ is } \mathbf{A} \Rightarrow \mathbf{B} \text{ is } \mathbf{Y}>) / fs(<\mathbf{X}, \mathbf{A}>) \qquad (3.11)$$

A fuzzy association rule is considered frequent if its confidence greater or equal to a fuzzy minimum confidence (*fminconf*) threshold specified by users. This means that the confidence must satisfy the condition below:

$$fc(\mathbf{X} \text{ is } \mathbf{A} \Rightarrow \mathbf{Y} \text{ is } \mathbf{B}) \geq fminconf.$$

Toán tử T-norm (⊗): there are various ways to choose T-norm operator [1] [2] [21] [47] for formula (3.6) such as:

- Min function: $a \otimes b = \min(a, b)$
- Normal multiplication: $a \otimes b = a.b$
- Limited multiplication: $a \otimes b = \max(0, a + b - 1)$
- Drastic multiplication: $a \otimes b = a$ (if b=1), $= b$ (if a=1), $= 0$ (if a, b < 1)
- Yager joint operator: $a \otimes b = 1 - \min[1, ((1-a)^w + (1-b)^w)^{1/w}]$ (with $w > 0$). If $w = 1$, it becomes limited multiplication. If w runs up to $+\infty$, it will develops into *min* function. If w decreases to 0, it becomes Drastic multiplication.

Based on experiments, we conclude that *min* function and *normal multiplication* are the two most preferable choices for T-norm operator because they are convenient to calculate support factors as well as can highlight the logical relations among fuzzy attributes in frequent fuzzy itemsets. The two following formulas (3.12) and (3.13) are derived from formula (3.6) by applying *min* function and *normal multiplication* respectively.

$$fs(<X,A>) = \frac{\sum_{v=1}^{m} \min\left\{\alpha_{x_1}(t_v[x_1]), \alpha_{x_2}(t_v[x_2]), \ldots, \alpha_{x_p}(t_v[x_p])\right\}}{|T|} \quad (3.12)$$

$$fs(<X,A>) = \frac{\sum_{v=1}^{m} \prod_{x_u \in X} \left\{\alpha_{x_u}(t_v[x_u])\right\}}{|T|} \quad (3.13)$$

Another reason for choosing *min* function and *algebraic multiplication* for T-norm operator is related to the question "*how we understand the meaning of the implication operator (*$\rightarrow$ or $\Rightarrow$*) in fuzzy logic theory?*". In the classical logic, the implication operator, used to link two clauses P and Q to form a compound clause P $\rightarrow$ Q, expresses the idea "if P then Q". This is a relatively complicated logical link because it is used to represent a *cause and effect* relation. While formalizing, we, however, consider the *truth* value of this relation as a regular combination of those of P and Q. This assumption may lead us to a misconception or a misunderstanding of this kind of relation [1].

In fuzzy logic theory, implication operator expresses a compound clause in the form of "if *u* is P then *v* is Q", in which, P and Q are two fuzzy sets on two universal domain U and V respectively. The *cause and effect* rule "if *u* is P then *v* is Q" is equivalent to that the pair (*u*, *v*) is a fuzzy set on the universal domain UxV. The fuzzy implication P $\rightarrow$ Q is considered a fuzzy set and we need to identify its membership function $m_{P\rightarrow Q}$ from membership functions m_P and m_Q of fuzzy sets P and Q. There are various researches around this issue. We relate herein several ways to determine membership function $m_{P\rightarrow Q}$ [1]:

- If adopting the idea of implication operator in classical logic theory, we have: $\forall$(u, v) $\in$ U x V: $m_{P\rightarrow Q}$(u, v) = $\oplus$(1- m_P, m_Q), in which, $\oplus$ is S-norm operator in fuzzy logic theory. If $\oplus$ is replaced with *max* function, we obtain the Dienes formula $m_{P\rightarrow Q}$(u, v) = max(1- m_P, m_Q). If $\oplus$ is replaced with *probability sum*, we receive the Mizumoto formula $m_{P\rightarrow Q}$(u, v) = 1- m_P + m_P.m_Q. And, if $\oplus$ is substituted by *limited multiplication*, we get the Lukaciewicz formula as

$m_{P\to Q}(u, v) = \min(1, 1- m_P + m_Q)$. In general, the $\oplus$ can be substituted by any valid function satisfying certain conditions of S-norm operator.

- Another way to interpret the meaning of this kind of relation is that the *truth* value of compound clause "if *u* is P then *v* is Q" increases iff the *truth* values of both antecedent and consequent are large. This means that $m_{P\to Q}(u, v) = \otimes(m_P, m_Q)$. If the $\otimes$ operator is substituted with *min* function, we receive the Mamdani formula $m_{P\to Q}(u, v) = \min(m_P, m_Q)$. Similarly, if $\otimes$ is replaced by *normal multiplication*, we obtain the formula $m_{P\to Q}(u, v) = m_P . m_Q$ [2].

Fuzzy association rule, in a sense, is a form of the fuzzy implication. Thus, it must comply with the above understandings. Although there are many combinations of m_P and m_Q to form the $m_{P\to Q}(u, v)$, the Mamdani formulas should be the most favorable ones. This is the main reason that influences our choice of *min* function and *algebraic multipication* for T-norm operator.

3.2.3 Algorithm for Fuzzy Association Rules Mining

The issue of discovering fuzzy association rules is usually decomposed into two following phases:

- Phase one: finding all possible frequent fuzzy itemset <**X**, **A**> from the input database, i.e. *fs*(<**X**, **A**>) ≥ *fminsup*.
- Phase two: generating all possible confident fuzzy association rules from the discovered frequent fuzzy itemsets above. This subproblem is relatively straightforward and less time-consuming comparing to the previous step. If <**X**, **A**> is a frequent fuzzy itemset, the rules we receive from <**X**, **A**> has the form of $X' \text{ is } A' \xrightarrow{fc} X \setminus X' \text{ is } A \setminus A'$, in which, **X'** and **A'** are non-empty subsets of **X** and **A** respectively. The inverse slash (i.e. \ sign) in the implication denotes the subtraction operator between two sets. *fc* is the fuzzy confidence factor of the rule and must meet the condition *fc* ≥ *fminconf*.

The inputs of the algorithm are a database **D** with attribute set **I** and record set **T**, and *fminsup* as well as *fminconf*.

The outputs of the algorithm are all possible confident fuzzy association rules.

Notation table:

Notations	Description
D	A relational or transactional database
I	Attribute set in **D**
T	Record set in **D**
$\mathbf{D_F}$	The output database after applying fuzzification over the original database **D**
$\mathbf{I_F}$	Set of fuzzy attributes in $\mathbf{D_F}$, each of them is attached with a fuzzy set. Each fuzzy set *f*, in turn, has a threshold w_f as used in formula (3.7)
$\mathbf{T_F}$	Set of records in $\mathbf{D_F}$, value of each record at a given fuzzy attribute is in [0, 1]
$\mathbf{C_k}$	Set of fuzzy k-itemset candidates
$\mathbf{F_k}$	Set of frequent fuzzy k-itemsets
F	Set of all possible frequent itemsets from database $\mathbf{D_F}$
fminsup	Fuzzy minimum support
fminconf	Fuzzy minimum confidence

Table 9 - Notations used in fuzzy association rules mining algorithm

The algorithm:

```
1   BEGIN
2   (D_F, I_F, T_F) = FuzzyMaterialization(D, I, T);
3   F_1 = Counting(D_F, I_F, T_F, fminsup);
4   k = 2;
5   while (F_{k-1} ≠ ∅) {
6         C_k = Join(F_{k-1});
7         C_k = Prune(C_k);
8         F_k = Checking(C_k, D_F, fminsup);
9         F = F ∪ F_k;
10        k = k + 1;
11  }
12  GenerateRules(F, fminconf);
13  END
```

Table 10 - Algorithm for mining fuzzy association rules

The algorithm in table 10 uses the following sub-programs:

- ($\mathbf{D_F}$, $\mathbf{I_F}$, $\mathbf{T_F}$) = **FuzzyMaterialization**(**D**, **I**, **T**): this function is to convert the original database **D** into the fuzzified database $\mathbf{D_F}$. Afterwards, **I** and **T** are also transformed to $\mathbf{I_F}$ and $\mathbf{T_F}$ respectively.

For example, with the database in table 8, after running this function, we will obtain:

$\mathbf{I_F}$ = {[*Age*, *Age_Young*] (1), [*Age*, *Age_Middle-aged*] (2), [*Age*, *Age_Old*] (3), [*Cholesterol*, *Cholesterol_Low*] (4), [*Cholesterol*, *Cholesterol_High*] (5), [*BloodSugar*, *BloodSugar_0*] (6), [*BloodSugar*, *BloodSugar_1*] (7), [*HeartDisease*, *HeartDisease_No*] (8), [*HeartDisease*, *HeartDisease_Yes*] (9)}

After converting, $\mathbf{I_F}$ contains 9 new fuzzy attributes comparing to 4 in **I**. Each fuzzy attribute is a pair including the name of original attribute and the name of corresponding fuzzy set and surrounded by square brackets. For instance, after fuzzifying the *Age* attribute, we receive three new fuzzy attributes [*Age*, *Age_Young*], [*Age*, *Age_Middle-aged*], and [*Age*, *Age_Old*].

In addition, the function **FuzzyMaterialization** also converts **T** into $\mathbf{T_F}$ as shown in the following table:

A	1	2	3	C	4	5	S	6	7	H	8	9
60	0.00	0.41	0.92	206	0.60	0.40	0	1	0	2	0	1
54	0.20	0.75	0.83	239	0.56	0.44	0	1	0	2	0	1
54	0.20	0.75	0.83	286	0.52	0.48	0	1	0	2	0	1
52	0.29	0.82	0.78	255	0.54	0.46	0	1	0	2	0	1
68	0.00	0.32	1.00	274	0.53	0.47	1	0	1	2	0	1
54	0.20	0.75	0.83	288	0.51	0.49	1	0	1	1	1	0
46	0.44	0.97	0.67	204	0.62	0.38	0	1	0	1	1	0
37	0.59	0.93	0.31	250	0.54	0.46	0	1	0	1	1	0
71	0.00	0.28	1.00	320	0.43	0.57	0	1	0	1	1	0
74	0.00	0.25	1.00	269	0.53	0.47	0	1	0	1	1	0
29	0.71	0.82	0.25	204	0.62	0.38	0	1	0	1	1	0
70	0.00	0.28	1.00	322	0.43	0.57	0	1	0	2	0	1
67	0.00	0.32	1.00	544	0.00	1.00	0	1	0	1	1	0

Table 11 - $\mathbf{T_F}$: Values of records at attributes after fuzzifying

Note that the characters **A**, **C**, **S**, and **H** in the table 11 are all the first character of *Age*, *Cholesterol*, *Sugar*, and *Heart* respectively.

Each fuzzy set *f* is accomanied by a threshold w_f, so only values that greater or equal to that threshold are taken into consideration. All other values are considered as 0. All gray cells in the table 11 indicates that theirs values are larger or equal to threshold (all thresholds in table 11 are 0.5). All values located in white-ground cells are equal to 0.

- $\mathbf{F_1}$ = **Counting**($\mathbf{D_F}$, $\mathbf{I_F}$, $\mathbf{T_F}$, ***fminsup***): this function is to generate $\mathbf{F_1}$, that is set of all frequent fuzzy 1-itemsets. All elements in $\mathbf{F_1}$ must have supports greater or equal to *fminsup*. For instance, if applying the *normal multiplication* for T-norm (⊗) operator in formula (3.6) and *fminsup* with 46%, we achieve the $\mathbf{F_1}$ that looks like the following table:

Fuzzy 1-itemsets	**Support**	**Is it frequent? *fminsup* = 46%**
{[*Age*, *Age_Young*]} (1)	10 %	No
{[*Age*, *Age_Middle-aged*]} (2)	45 %	No
{[*Age*, *Age_Old*]} (3)	76 %	Yes
{[*Serum cholesterol*, *Cholesterol_Low*]} (4)	43 %	No
{[*Serum cholesterol*, *Cholesterol_High*]} (5)	16 %	No
{[*BloodSugar*, *BloodSugar_0*]} (6)	85 %	Yes
{[*BloodSugar*, *BloodSugar_1*]} (7)	15 %	No
{[*HeartDisease*, *HeartDisease_No*]} (8)	54 %	Yes
{[*HeartDisease*, *HeartDisease_Yes*]} (9)	46 %	Yes

Table 12 - C_1: set of candidate 1-itemsets

$$\mathbf{F_1} = \{\{3\}, \{6\}, \{8\}, \{9\}\}$$

- $\mathbf{C_k}$ = **Join**($\mathbf{F_{k-1}}$): this function is to produce the set of all fuzzy candidate k-itemsets ($\mathbf{C_k}$) based on the set of frequent fuzzy (k-1)-itemsets ($\mathbf{F_{k-1}}$) discovered in the previous step. The following SQL statement indicates how elements in $\mathbf{F_{k-1}}$ are combined to form candidate k-itemsets.

  INSERT INTO C_k
  SELECT $p.i_1, p.i_2, \ldots, p.i_{k-1}, q.i_{k-1}$
  FROM L_{k-1} p, L_{k-1} q
  WHERE $p.i_1 = q.i_1, \ldots, p.i_{k-2} = q.i_{k-2}, p.i_{k-1} < q.i_{k-1}$ AND $p.i_{k-1}.o \neq q.i_{k-1}.o$;

  In which, $p.i_j$ and $q.i_j$ are index number of j^{th} fuzzy attributes in itemsets p and q respectively. $p.i_j.o$ and $q.i_j.o$ are the index number of original attribute. Two fuzzy attributes sharing a common original attribute must not exist in the same fuzzy itemset. For example, after running the above SQL command, we obtain $\mathbf{C_2}$ = {{3, 6}, {3, 8}, {3, 9}, {6, 8}, {6, 9}}. The 2-itemset {8, 9} is invalid because its two fuzzy attributes are derived from a common attribute *HeartDisease*.

- $\mathbf{C_k}$ = **Prune**($\mathbf{C_k}$): this function helps us to prune any unnecessary candidate k-itemset in $\mathbf{C_k}$ thanks to the downward closure property "*all subsets of a frequent itemset are also frequent, and any superset of a non-frequent itemset will be not frequent*". To evaluate the usefulness of any k-itemste in $\mathbf{C_k}$, the **Prune** function must make sure that all (k-1)-subsets of $\mathbf{C_k}$ are present in $\mathbf{F_{k-1}}$.

  For instance, after pruning the $\mathbf{C_2}$ = {{3, 6}, {3, 8}, {3, 9}, {6, 8}, {6, 9}}.

- $\mathbf{F_k}$ = **Checking**($\mathbf{C_k}$, $\mathbf{D_F}$, ***fminsup***): this function first scans over the whole transactions in datatabases to update support factors for candidate itemsets in $\mathbf{C_k}$. Afterwards, **Checking** eliminates any infrequent candidate itemset, i.e. whose support is smaller than ***fminsup***. All frequent itemsets are retained and put into $\mathbf{F_k}$.

  After running $\mathbf{F_2}$ = **Checking**($\mathbf{C_2}$, $\mathbf{D_F}$, **46%**), we receive $\mathbf{F_2}$ = {{3,6}, {6,8}}. The following table displays the detailed information.

Candidate 2-itemset	Support factor	Is it frequent?
{3, 6}	62 %	Yes
{3, 8}	35 %	No
{3, 9}	41 %	No
{6, 8}	46 %	Yes
{6, 9}	38 %	No

Table 13 - F_2: set of frequent 2-itemsets

- **GenerateRules**(**F**, ***fminconf***): this function generates all possible confident fuzzy association rules from the set of all frequent fuzzy itemsets **F**.

  With the above example, after finishing the first phase (finding all possible frequent itemsets), we get $\mathbf{F} = \mathbf{F_1} \cup \mathbf{F_2}$ = {{3}, {6}, {8}, {9}, {3,6}, {6,8}} ($\mathbf{F_3}$ is not created because $\mathbf{C_3}$ is empty). The following table lists discovered fuzzy association rules.

No.	Fuzzy association rules or 1-itemsets	Support	Confidence
1	Old people	76 %	
2	Blood sugar ≤ 120 mg/ml	85 %	
3	Not suffer from heart disease	54 %	
4	Suffer from heart disease	46 %	
5	Old people => Blood sugar ≤ 120 mg/ml	62 %	82 %
6	Blood sugar ≤ 120 mg/ml => Old people	62 %	73 %
7	Blood sugar ≤ 120 mg/ml => Not suffer from heart disease	46 %	54 %
8	Not suffer from heart disease => Blood sugar ≤ 120 mg/ml	46 %	85 %

Table 14 - Fuzzy association rules generated from database in table 8

The minimum confidence is 70%, so the 7th rule is rejected.

3.2.4 Relation between Fuzzy Association Rules and Quantitative Association Rules

According to the formula (3.7), the membership function of each fuzzy set *f* is attached with a threshold w_f. Based on this threshold, we can defuzzify to convert association rule into another form similar to quantitative one.

For example, the fuzzy rule "*Old people => Blood sugar ≤ 120 mg/ml*, with support 62% and confidence 82%" in the table 14 should be changed to the rule "*Age ≥ 46 => Blood sugar ≤ 120 mg/ml*, with support 62% and confidence 82%". We see the minimum value of attribute [*Age*, *Age_Old*] that greater or equal to $w_{\text{Age\_Old}}$ (= 0.5) is 0.67. The age corresponding to the fuzzy value 0.67 is 46, so any person whose age larger or equal to 46 will have fuzzy value greater or equal to 0.67. Therefore, we substitute "*Age_Old*" by "*Age ≥ 46*". Similarly, we can change any fuzzy association rule to quantitative one.

Because almost all fuzzy membership functions have smooth graph or their derivatives change the sign (from positive to nagative and vice versa), the defuzzification is relatively simple.

3.2.5 Experiments and Conclusions

- Experiments by varying the database size (increase or decrease the number of records) to measure the mining time.
- Experiments by changing the *fminsup* and *fminconf*

- Experiments in changing the thresholds associated with fuzzy sets.
- Experiments by changing the choice of T-norm operator (*min* function and *normal multiplication*)
- Experiment the transformation from fuzzy association rules to quantitative ones.

Chapter 4. Parallel Mining of Fuzzy Association Rules

One of the most essential and time-consuming tasks in association rules mining is finding all possible frequent itemsets from immense volumes of data. It needs much CPU time (CPU-bound) and I/O operation (I/O-bound). Thus, researchers have been trying their best to improve the existing algorithms or devise new ones in order to speed up the whole mining process [11] [13] [20] [24] [30] [35]. Most of these algorithms are sequential and work efficiently upon small or medium databases (the sizes of databases are recognized based on their number of attributes and records). However, they lose their performance and expose some disadvantages while working with extremely large databases (usually hundreds of megabytes or more) due to the limitations in the processor's speed as well as the capacity of internal memory of a single computer.

Fortunately, with the explosive development in hardware industry, high performance computing systems are introduced to the market. This has opened up an opportunity for a new research direction in data mining community. Since 1995, researchers continually devise efficient parallel and distributed algorithms for the issue of association rules mining [5] [12] [18] [26] [31] [32] [34]. These algorithms are diverse because of their tight dependences upon architectures of various parallel computing systems.

In the first section of this chapter, I coarsely present previous parallel algorithms that are successfully experimented on real databases. In the next section, I would like to recommend a novel parallel algorithm for mining fuzzy association rules. It has been experimented on a Linux-based PC-Cluster system using Message Passing Interface standard (MPI) [27] [28] [29] and returns optimistic results. This algorithm is relatively optimal because it strongly reduces the data communication and synchronization among processors. However, it can only mine the fuzzy or quantitative association rules as well as suite for relational rather than transactional databases.

4.1 Several Previously Proposed Parallel Algorithms

In this section, I would like to restate some existing parallel algorithms. They are all designed to run on shared-nothing parallel systems that have the following characteristics:

- A shared-nothing system has **N** processors. Each processor $\mathbf{P^i}$ has its own internal (RAM) and external (hard disk) memory.
- **N** processors can communicate with each other thanks to a high-speed network using message passing mechanism.

4.1.1 Count Distribution Algorithm

The *Count Distribution* approach proposed in [34] is an Apriori-like algorithm [35]. In this algorithm, **N** denotes the number of processors, $\mathbf{P^i}$ stand for i^{th} processor, and $\mathbf{D^i}$ is the data partition for processor $\mathbf{P^i}$ (the original database **D** is divided into **N** equal partitions, each of them is placed on the hard disk of a processor. The algorithm consists of the following steps:

- (1) Pass one (k = 1): the first pass is special. All processors in the system will receive the set of frequent 1-itemsets, i.e. $\mathbf{L_1}$.
- (2) Pass two: for all k > 1, the algorithm repeatedly perform the following steps:
    - (2.1): Each processor $\mathbf{P^i}$ generates the complete $\mathbf{C_k}$, using the complete set of frequent itemsets $\mathbf{L_{k-1}}$ created at the pass k-1. Observe that since each processor has the identical $\mathbf{L_{k-1}}$, they will be generating identical $\mathbf{C_k}$.
    - (2.2): Processor $\mathbf{P^i}$ makes a pass over its data partition $\mathbf{D^i}$ and develops local support counts for candidate itemsets in $\mathbf{C_k}$. This is the parallel step. This means that all processors can independently scan over its own data partition.

- (2.3): Processor $\mathbf{P^i}$ exchanges local $\mathbf{C_k}$ support counts with all other processors to develop the global $\mathbf{C_k}$ support counts. Processors are forced to synchronize in this step.
- (2.4): Each processor $\mathbf{P^i}$ now computes $\mathbf{L_k}$ from $\mathbf{C_k}$ based on *minsup* parameter.
- (2.5): Each processor $\mathbf{P^i}$ independently makes the decision to terminate or continue the next loop. The decision will be identical as the processors all have identical $\mathbf{L_k}$.

The figure below illustrates the Count Distribution algorithm.

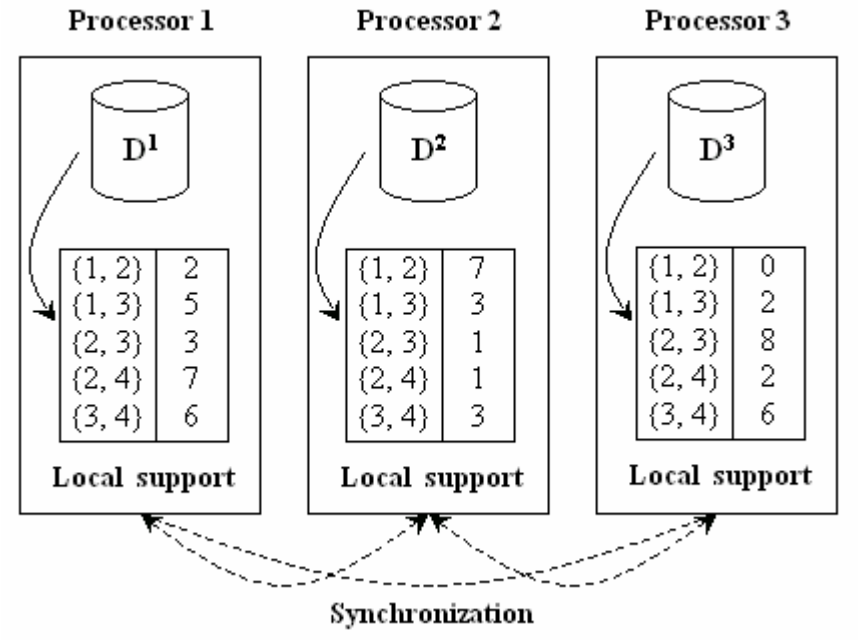

Figure 7 - Count distribution algorithm on a 3-processor parallel system

4.1.2 Data Distribution Algorithm

The attractive feature of the Count Distribution algorithm is that no data tuples are exchanged between processors – only support counts are exchanged. Thus, processors can operate independently and asynchronously while reading the data. However, the disadvantage is that this algorithm does not exploit the aggregate memory of the system effectively. Suppose that each processor has memory of size |**M**|. The number of candidate itemsets that can be counted in one pass is determined by |**M**|. As we increase the number of processors from 1 to **N**, the

system has **N** x |**M**| total memory, but we still count the same number of candidate itemsets in one pass, as each processor is counting for identical $\mathbf{C_k}$.

The Data distribution algorithm is designed to exploit better total system's memory as the number of processors is increased. In this algorithm, each processor counts mutually exclusive candidate itemsets. Thus, as the number of processors is increased, a larger number of candidate itemsets can be counted in a pass. The downside of this algorithm is that every processor must broadcast its local data to all other processors in every pass. Therefore, this algorithm can become feasible only on a machine with very fast and stable communication.

The Data distribution is also based on the Apriori. In this algorithm, **N** denotes the number of processors, $\mathbf{P^i}$ is the i^{th} processor, and $\mathbf{D^i}$ is the data partition for processor $\mathbf{P^i}$ (the original database **D** is divided into **N** equal partitions, each of them is placed on the hard disk of a processor. The algorithm includes the following steps:

- (1) Pass one ($k = 1$): The same Count distribution algorithm.
- (2) Pass two (for every $k > 1$):
    - (2.1): Processor $\mathbf{P^i}$ generates $\mathbf{C_k}$ from $\mathbf{L_{k-1}}$. It retains only $1/\mathbf{N}^{th}$ of the itemsets forming the candidate subset $\mathbf{C_k^i}$ that it will count. Which $1/\mathbf{N}$ itemsets are retained is determined by the processor identifier and can be computed without communicating with other processors. In real implementation, itemsets are assigned in a round-robin fashion. The $\mathbf{C_k^i}$ sets are all disjoint and union of all $\mathbf{C_k^i}$ sets is the original $\mathbf{C_k}$ (i.e. $\mathbf{C_k^i} \cap \mathbf{C_k^j} = \varnothing$ (where i ≠ j) and $\bigcup_{i=1}^{N} C_k^i = C_k$).
    - (2.2): Processor $\mathbf{P^i}$ develops support counts for itemsets in its local candidate set $\mathbf{C_k^i}$ using both local data pages and data page received from other processors.
    - (2.3): At the end of the pass over the data, each processor $\mathbf{P^i}$ calculates $\mathbf{L_k^i}$ using the local $\mathbf{C_k^i}$. Again, all $\mathbf{L_k^i}$ sets are disjoint and

the union of all $\mathbf{L_k^i}$ sets is $\mathbf{L_k}$ (i.e. $\mathbf{L_k^i} \cap \mathbf{L_k^j} = \varnothing$ where i ≠ j and $\bigcup_{i=1}^{N} L_k^i = L_k$).

- (2.4): Processors exchanges $\mathbf{L_k^i}$ so that every processor has the complete $\mathbf{L_k}$ for generating $\mathbf{C_{k+1}}$ for the next pass. This step requires processors to synchronize. Having obtained the complete $\mathbf{L_k}$, each processor can independently (but identically) decide whether to terminate or continue on to the next loop.

The following figure depicts this algorithm:

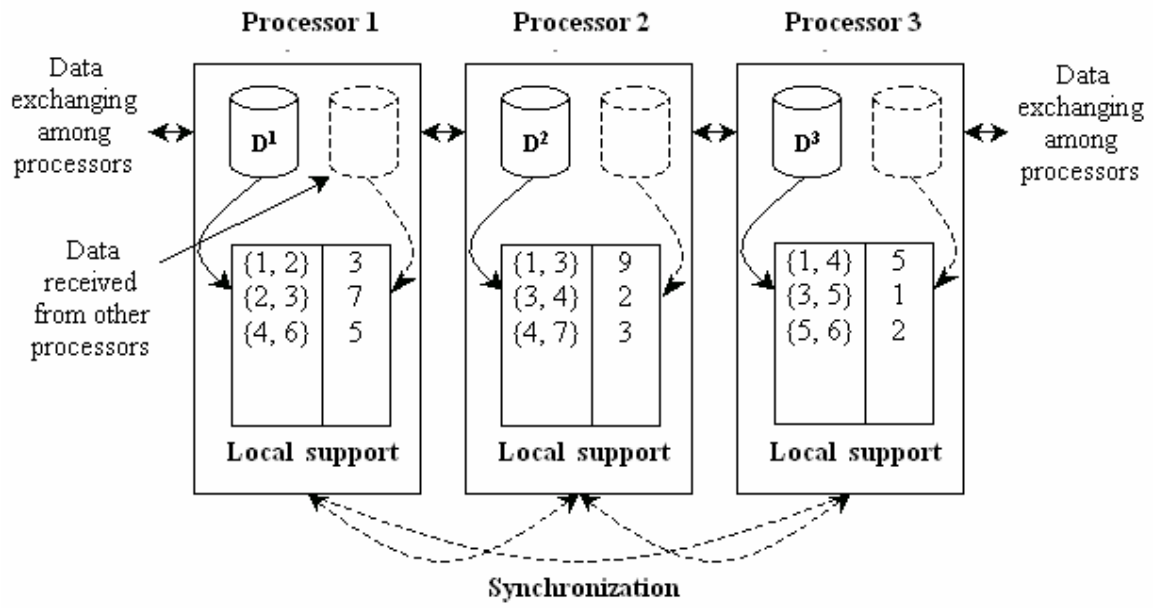

Figure 8 - Data distribution algorithm on a 3-processor parallel system

4.1.3 Candidate Distribution Algorithm

Both Count and Data distribution algorithms require processors to synchronize at the end of a pass to exchange support counts or frequent itemsets respectively. If the workload is not perfectly balanced, this can cause all the processors to wait for whichever processor finishes last in every pass. The Candidate distribution algorithm attempts to do away with the dependence between processors so that they may proceed independently without synchronizing at the end of every pass.

In some pass *l*, where *l* is heuristically determined, this algorithm divides the frequent itemsets $\mathbf{L_{l-1}}$ between processors in such a way that a processor $\mathbf{P^i}$ can

generate a unique $\mathbf{C_m^i}$ ($m \geq l$) independent of all other processors ($\mathbf{C_m^i} \cap \mathbf{C_m^j} = \varnothing$, $\forall i \neq j$). At the same time, data is selectively replicated so that a processor can count candidates in $\mathbf{C_m^i}$ independent of all other processors. This choice of the redistribution pass is a tradeoff between decoupling processor dependence as soon as possible and waiting until the itemsets become more easily and equitably partitionable. The partitioning algorithm exploits the semantics of the Apriori candidate generation procedure described before.

After this candidate distribution, the only dependence that a processor has on other processors is for pruning the local candidate set during the prune step of candidate generation. However, a processor does not wait for the complete pruning information to arrive from all other processors. During the prune step of candidate generation, it prunes the candidate set as much as possible using whatever information has arrived, and opportunistically starts counting the candidates. The late arriving pruning information can be used in subsequent passes. The algorithm is described below:

- (1) Pass one ($k < l$): Use either Count or Data distribution algorithm.
- (2) Pass two ($k = l$):
    - (2.1): Partition $\mathbf{L_{k-1}}$ among the **N** processors such that $\mathbf{L_{k-1}}$ sets are "well balanced". We discuss below how this partitioning is done. Record with each frequent itemset in $\mathbf{L_{k-1}}$ which processor has been assigned this itemset. This partitioning is identically done in parallel by each processor.
    - (2.2): Processor $\mathbf{P^i}$ produces $\mathbf{C_k^i}$, logically using only the $\mathbf{L_{k-1}}$ partition assigned to it. Note that P^i still has access to the complete $\mathbf{L_{k-1}}$, and hence can use standard pruning while generating $\mathbf{C_k^i}$ in this pass.
    - (2.3): $\mathbf{P^i}$ develops global counts for candidates in $\mathbf{C_k^i}$ and the database is repartitioned into $\mathbf{DR^i}$ at the same time. The details of this step are given below.
    - (2.4): After $\mathbf{P^i}$ has processed all its local data and any data received from all other processors, it posts **N**-1 asynchronous receive buffers

to receive $\mathbf{L_k^j}$ from all other processors. These $\mathbf{L_k^j}$ are needed for pruning $\mathbf{C_{k+1}^i}$ in the prune step of candidate generation.

- (2.5): Processor $\mathbf{P^i}$ computes $\mathbf{L_k^i}$ from $\mathbf{C_k^i}$ and asynchronously broadcast it to the other **N**-1 processors using **N**-1 asynchronous sends.

- (3) Pass ($k > l$):
  - (3.1): Processor $\mathbf{P^i}$ collects all frequent itemsets that have sent to it by other processors. They are used in the pruning step of the candidate generation, but not the join step. Itemsets received from processor j could be of length k-1, smaller than k-1 (slower processor), or greater than k-1 (faster processor). $\mathbf{P^i}$ keeps track for each processor $\mathbf{P^j}$ the largest size of the frequent itemsets sent by it. Receive buffers for the frequent itemsets are reposted after processing.
  - (3.2): $\mathbf{P^i}$ creates $\mathbf{C_k^i}$ using the local $\mathbf{L_{k-1}^i}$. Now it can happen that $\mathbf{P^i}$ has not received $\mathbf{L_{k-1}^j}$ from all other processors, so $\mathbf{P^i}$ needs to be careful at the time of pruning. It needs to distinguish an itemset (a k-1 long subset of a candidate itemset) which is not present in any of $\mathbf{L_{k-1}^j}$ from an itemset that is present in some $\mathbf{L_{k-1}^j}$ but this set has not yet been received by processor $\mathbf{P^i}$. It does so by probing $\mathbf{L_{l-1}}$ (remember that repartitioning took place in pass l) using the l-1 long prefix of the itemset in question, finding the processor reponsible for it, and checking if $\mathbf{L_{k-1}^j}$ has been received from this processor.
  - (3.3): $\mathbf{P^i}$ makes a pass over $\mathbf{DR^i}$ and counts $\mathbf{C_k^i}$. It then computes $\mathbf{L_k^i}$ and asynchronously broadcast $\mathbf{L_k^i}$ to every other processor using **N**-1 asynchronous sends.

***Partitioning*:** We motivate the algorithm for partitioning $\mathbf{L_k}$ by an example. Let $\mathbf{L_3}$ be {ABC, ABD, ABE, ACD, ACE, BCD, BCE, BDE, CDE}. Then $\mathbf{L_4}$ = {ABCD, ABCE, ABDE, ACDE, BCDE}, $\mathbf{L_5}$ = {ABCDE} và $\mathbf{L_6} = \varnothing$. Consider ε = {ABC, ABD, ABE} whose members all have the common prefix AB. Not that the candidates ABCD, ABCE, ABDE, and ABCDE also have the prefix AB. The

Apriori candidate generation procedure generates these candidates by joining only the items in ε.

Therefore, assuming that the items in the itemsets are lexicographically ordered, we can partition the itemsets in $\mathbf{L_k}$ based on common k-1 long prefixes. By ensuing that no parition is assigned to more than one processor, we have ensured that each processor can create candidates independently (ignoring the prune step). Suppose we also repartition the database in such a way that any tuple that supports an itemset contained in any of the $\mathbf{L_k}$ partitions assigned to a processor is copied to the local disk of that processor. The processor can then proceed entire synchronization.

The actual algorithm is more involved because of two reasons. A processor may have to obtain frequent itemsets computed by other processors for the prune step of the candidate generation. In the example above, the processor assigned the set ε has to know whether BCDE is frequent to be able to decide whether to prune the candidate ABCDE, but the set with prefix BC may have been assigned to a different processor. The other problem is that we need to balance load across processors.

4.1.4 Algorithm for Parallel Generation of Association Rules

To generate rules, for every large itemset *l*, we find all non-empty subsets of *l*. For every such subset *a*, we output a rule of the form $a \Rightarrow (l - a)$ if the ratio of $s(l)$ to $s(a)$ is at least *minconf*. We consider all subsets of *l* to generate rules with multiple consequents. Since the large itemsets are store in hash tables, the support counts for the subset itemsets can be found efficiently.

We can improve the above procedure by generating the subsets of a large itemset in a recursive depth-first fashion. For example, given an itemset ABDC, we first consider the subset ABC, then AB, etc. Then if a subset *a* of a large itemset *l* does not generate a rule, the subsets of *a* need not be considered for generating rules using *l*. For example, if ABC => D does not have adequate confidence, we need not check whether AB => CD holds. We do not miss any rules because the support of any subset *a'* of *a* must be as great as the support of *a*. Therefore, the confidence of the rule $a' \Rightarrow (l - a')$ cannot be more than the confidence of $a \Rightarrow (l$

– *a*). Hence, if *a* did not yield a rule involving all the items in *l* with *a* as the antecedent, neither will *a'*.

The sequential algorithm [35] for generating association rules is shown in the table below:

```
// Simple algorithm
Forall frequent itemset l_k, k > 1 do
      Call gen_rules(l_k, l_k);

// The gen_rules generates all valid rules α => (l - α), for all α ⊂ a_m
Procedure gen_rules(l_k : frequent k-itemset, a_m : frequent m-itemset)
Begin
1     A = {(m-1)-itemsets a_m-1 | a_m-1 ⊂ a_m}
2     Forall a_m-1 ∈ A do begin
3           conf = s(l_k)/s(a_m-1);
4           if (conf ≥ minconf) then begin
5                 output the rule a_m-1 => (l_k – a_m-1);
6                 if (m – 1 > 1) then
7                       Call gen_rules(l_k, a_m-1);
8           end
9     end
End
```

Table 15 - The sequential algorithm for generating association rules

Generating rules in parallel simply involves partitioning the set of all frequent itemsets among the processors. Each processor then generates rules for its partition only using the algorithm above. Since the number of rules that can be generated from an itemset is sensitive to the itemset's size, we attempt equitable balancing by partitioning the itemsets of each length equally across the processors.

Note that in the calculation of the confidence of a rule, a processor may need to examine the support of an itemset for which it is not responsible. For this reason, each processor must have access to all the frequent itemsets before rule generation can begin. This is not a problem for the Count and Data distribution algorithms because at the end of the last pass, all the processors have all the frequent itemsets. In the Candidate distribution algorithm, fast processors may need to wait until slower processors have discovered and transmitted all of their frequent itemsets. For this reason and because the rule generation step is relatively cheap, it may be better in the Candidate distribution algorithm to simply discover the frequent

itemsets and generate the rules off-line, possibly on a serial processor. This would allow processors to be freed to run other jobs as soon as they are done finding frequent itemsets, even while other processors in the system are still working.

4.1.5 Other Parallel Algorithms

In addition to four above algorithms, we also list herein some of other parallel algorithms for mining association rules.

The Intelligent Data Distribution algorithm [12] was proposed based on the normal data distribution with an additional improvement in data communication among processors. Instead of data transmission between every pair of processors, this algorithm transmit data based on a logical ring set up at the startup time of the algorithm.

The Multiple Local Frequent Pattern Tree (MLFPT) [31] was derived from the FP-growth algorithm. This algorithm needs no candidate generation and can reduce the number of passes over the database. Furthermore, it investigates the load balancing among processors.

Another parallel algorithm working on shared-everything systems (normally SMP systems) was proposed by [26].

4.2 A New Parallel Algorithm for Fuzzy Association Rules Mining

Almost all known parallel algorithms, more or less, need the data communication and synchronization among processors. This leads to an additional complexity in real implementations of these algorithms. Hence, they are not considered to be “ideal” parallel computing problems. Based on the approach in fuzzy association rules mentioned above, I would like to suggest a new parallel algorithm for mining this kind of rule. It is ideal that little communication needs to be taken place during the processing time. Data communication is made only twice - one at the startup for dividing and delivering fuzzy attributes among processors, and one for rules gathering as the algorithm finishes.

4.2.1 Our Approach

According to the fuzzy association rules mining algorithm mentioned in chapter three, each attribute i_u in **I** is associated with a set of fuzzy sets F_{i_u} as follows:

$$F_{i_u} = \left\{ f_{i_u}^1, f_{i_u}^2, ..., f_{i_u}^k \right\}$$

For example, in the sample database in table 8, we have:

$$F_{i_1} = \mathrm{F_{Age}} = \{Age\_Young, Age\_Middle\text{-}aged, Age\_Old\} \text{ (with } k = 3)$$

$$F_{i_2} = \mathrm{F_{Serum\_cholesterol}} = \{Cholesterol\_Low, Cholesterol\_High\} \text{ (with } k = 2)$$

The fuzzy association rule has the form of:

$$\mathbf{X} \text{ is } \mathbf{A} \Rightarrow \mathbf{Y} \text{ is } \mathbf{B}.$$

Where:

- $\mathbf{X}, \mathbf{Y} \subseteq \mathbf{I}$ are itemsets. $\mathbf{X} = \{x_1, x_2, \ldots, x_p\}$ ($x_i \neq x_j$ if $i \neq j$) and $\mathbf{Y} = \{y_1, y_2, \ldots, y_q\}$ ($y_i \neq y_j$ if $i \neq j$).
- $\mathbf{A} = \{f_{x1}, f_{x2}, \ldots, f_{xp}\}$, $\mathbf{B} = \{f_{y1}, f_{y2}, \ldots, f_{yq}\}$ are sets of fuzzy sets corresponding to attributes in **X** and **Y**. $f_{xi} \in F_{xi}$ và $f_{yj} \in F_{yj}$.

Each fuzzy attribute is a pair of attribute name accompanied by fuzzy set name. For instance, with **I** = {*Age*, *SerumCholesterol*, *BloodSugar*, *HeartDisease*}, we now have the set of fuzzy attributes $\mathbf{I_F}$ as:

$\mathbf{I_F}$ = {[*Age*, *Age_Young*] (1), [*Age*, *Age_Middle-aged*] (2), [*Age*, *Age_Old*] (3), [*Cholesterol*, *Cholesterol_Low*] (4), [*Cholesterol*, *Cholesterol_High*] (5), [*BloodSugar*, *BloodSugar_0*] (6), [*BloodSugar*, *BloodSugar_1*] (7), [*HeartDisease*, *HeartDisease_No*] (8), [*HeartDisease*, *HeartDisease_Yes*] (9)}

Table 16 - Set of fuzzy attributes received after being fuzzified the database in table 8

After being fuzzified, $\mathbf{I_F}$ consists of 9 fuzzy attributes comparing to 4 in **I**. All values in at fuzzy attributes in record set $\mathbf{T_F}$ is now belonging to the range [0, 1]. The objective of the mining issue is to discover all possible confident fuzzy association rules.

We totally perceive that any fuzzy association rule (both antecedent and consequent) never contains two fuzzy attributes that share a common original attribute in **I**. For example, the rules such "*Age_Old* AND *Cholesterol_High* And *Age_Young* => *HeartDisease_Yes*" or "*BloodSugar* > *120mg/ml* AND *HeartDisease_No* => *HeartDisease_Yes*" are invalid because the former contains both *Age_Old* and *Age_Young* (derived from *Age*), and the latter encompasses *HeartDisease_No* as well as *HeartDisease_Yes* (derived from *HeartDisease*).

There are two chief reasons for the above supposition. First, fuzzy attributes sharing a common original attribute are usually mutually exclusive in meaning so that they will largely decrease the support of rules in which they are contained together. For example, the *Age_Old* is opposite in semantics with *Age_Young* because no person in the world is "both *young* and *old*". Second, such rule is not worthwhile and carries little meaning.

Thus, we can conclude that all fuzzy attributes in the same rule are independent in that there is no pair of fuzzy attribute whose original attribute is the same. This observation is the foundation of our new parallel algorithm.

I will coarsely describe our idea via a simple example. Suppose that we will run our algorithm on the database in table 8 and on a 6-processor parallel system. We need to divide the set of fuzzy attributes $\mathbf{I_F}$ among processors so that processors can operate in parallel and independently as follows.

The processor $\mathbf{P^1}$:

$$\mathbf{I_F}^1 = \{[Age, Age\_Young]\ (1), [Cholesterol, Cholesterol\_Low]\ (4),$$
$$[BloodSugar, BloodSugar\_0]\ (6), [BloodSugar, BloodSugar\_1]\ (7),$$
$$[HeartDisease, HeartDisease\_No]\ (8), [HeartDisease, HeartDisease\_Yes]\ (9)\}$$
$$= \{1, 4, 6, 7, 8, 9\}$$

The processor $\mathbf{P^2}$: $\mathbf{I_F}^2 = \{1, 5, 6, 7, 8, 9\}$

The processor $\mathbf{P^3}$: $\mathbf{I_F}^3 = \{2, 4, 6, 7, 8, 9\}$

The processor $\mathbf{P^4}$: $\mathbf{I_F}^4 = \{2, 5, 6, 7, 8, 9\}$

The processor $\mathbf{P^5}$: $\mathbf{I_F}^5 = \{3, 4, 6, 7, 8, 9\}$

The processor $\mathbf{P^6}$: $\mathbf{I_F}^6 = \{3, 5, 6, 7, 8, 9\}$

We divide the $\mathbf{I_F}$ based on the first two attributes *Age* and *Cholesterol*. The 9 initial fuzzy attributes are now distributed among 6 processors and each processor receive 6 fuzzy attributes. This division is "ideal" because the number of processors (6) is equal to the multiplication of number fuzzy sets associated with attribute *Age* (3) and number of fuzzy sets associated with attribute *Cholesterol* (2) (i.e. 6 = 3*2).

The optimal division is where we could equally disperse fuzzy attributes to all processors in the system. In the case of being unable to obtain a optimal division, we will use "the most reasonable" one. This means that several processors are in idle state while others work hard. I would like to present an algorithm used for fuzzy attributes division. It first tries to find the optimal solution, if not it will return "the most reasonable" one. The division algorithm is formally described below:

Given a database **D** with $\mathbf{I} = \{i_1, i_2, \ldots, i_n\}$ is set of *n* attributes, and $\mathbf{T} = \{t_1, t_2, \ldots, t_m\}$ is set of *m* records. After being fuzzified, **D**, **I**, and **T** are converted into $\mathbf{D_F}$, $\mathbf{I_F}$, and $\mathbf{T_F}$ respectively.

$\mathbf{I_F} = \{[i_1, f_{i1}^{1}], \ldots, [i_1, f_{i1}^{k1}], [i_2, f_{i2}^{1}], \ldots, [i_2, f_{i2}^{k2}], \ldots, [i_n, f_{in}^{1}], \ldots, [i_n, f_{in}^{kn}]\}$.

Where, f_{ij}^{u} and k_j are the u^{th} fuzzy set and the number of fuzzy sets associated with attribute i_j. For example, the database in table 8, we have **I** = {*Age*, *SerumCholesterol*, *BloodSugar*, *HeartDisease*} and after converting we receive $\mathbf{I_F}$ as shown in table 16. In this case, $k_1 = 3$, $k_2 = 2$, $k_3 = 2$, $k_4 = 2$ are numbers of fuzzy sets associated with original attributes in **I**.

Let $\mathbf{FN} = \{k_1\} \cup \{k_2\} \cup \ldots \cup \{k_n\} = \{s_1, s_2, \ldots, s_v\}$ ($v \leq n$ as the may be pairs such as (k_i, k_j) that equal) and **N** is the number of processors in the system. The division algorithm is changed to the problem stated as: *Find the non-empty subset **Fn** of **FN** such that the multiplication among elements in **Fn** is equal to **N** (this is the optimal solution). In the case of being unable to obtain the optimal solution, the algorithm will return "the most reasonable" solution. This means that the multiplication among elements in **Fn** is a lower approximation of **N**.*

This algorithm is designed according to *backtracking* strategy, it is whether recursive or not.

The division algorithm:

```
BOOLEAN Subset(FN, N, Idx)
1       k = 1;
2       Idx[1] = 0;
3       S = 0;
4       While (k > 0) {
5               Idx[k]++;
6               if (Idx[k] <= sizeof(FN)) {
7                       if (S * FN[Idx[k]] <= N) {
8                               if (S * FN[Idx[k]] == N)
9                                       return TRUE;
10                              else {
11                                      S *= FN[Idx[k]];
12                                      Idx[k + 1] = Idx[k];
13                                      k++;
14                              }
15                      }
16              } else {
17                      k--;
18                      S /= FN[Idx[k]];
19              }
20      }
21      return FALSE;

FindSubset(FN, N, Idx, Fn)
1       for (n = N; n > 0; n--)
2               If (Subset(FN, n, Idx)) {
3                       Fn = {FN[i] | i ∈ Idx}
4                       return;
5               }
```

Table 17 - Fuzzy attributes dividing algorithm among processors

In the above example, the **Fn** is {3, 2}. The above algorithm surely to return "the most reasonable" solution in the case not receiving the optimal one since the **FindSubset** will gradually decrease *n* to search for lower approximation space of **N**.

4.2.2 The New Algorithm

Inputs: The database **D** with the attribute set **I** and the record set **T,** the number of processors is refered to as **N,** The *minsup* and *minconf* indicate the minimum support and minimum confidence threshold respectively.

Outputs: All possible confident fuzzy association rules.

The parallel algorithm for mining fuzzy association rules includes the following steps:

- (1): Converting the database **D**, attribute set **I,** and record set **T** into $\mathbf{D_F}$, $\mathbf{I_F}$, and $\mathbf{T_F}$ respectively. This is fuzzification process.
- (2): Using the algorithm in table 17 for fuzzy attributes scattering among **N** processors in the system.
- (3): Each processor $\mathbf{P^i}$ use the sequential algorithm in table 10 to mine frequent itemsets and algorithm in table 15 to generate confident fuzzy association rules.
- (4): Collecting discovered rules from all processors in the system.

This algorithm is successfully applied to not only fuzzy association rules but quantitative and categorical ones.

4.3 Experiments and Conclusions

- Experimenting while varying number of attributes and number of records to measure mining time.
- Experimenting while increasing and decreasing the number of processors in the system

Chapter 5. Conclusions

5.1 Achievements throughout the dissertation

Being based on the previous research outputs in the field of data mining, the dissertation not only sums up the most noticeable features in this research direction but also presents several new proposals. We summarize herein the chief achievements throughout the thesis.

In the first chapter, the thesis outlines the overall picture of data mining - some descriptive definitions as well as its motivation. This section also generally mentions the dominant techniques that are applied for subproblems such as clustering, classification/prediction, association rules mining, etc. Finnaly, the chapter enumerates the recently focused research trends in this research domain.

The chapter two formally describes the issue of *association rules mining* proposed by R. Agrawal in 1993. The most fundamental concepts are also given such as itemset, frequent itemset, confident rule, etc. Moreover, this chapter depicts some most interesting research topics that are concentrated in recent years such as fuzzy association rules, parallel mining of association rules, etc. The core target of this chapter is to present the background of association rules mining to readers. A thorough understanding of this chapter will be very convenient for studying the next two chapters.

On the foundation of the previous proposals of [4] [9] [38] [39], chapter three present the problem of quantitative & categorical association rules mining together with its advantages and disadvantages. However, the focused target of this chapter is to deeply study a special kind of rule - the fuzzy association rule. This type of rule is much more flexible, readable, and intuitive comparing to the elementary kind of rule described in the previous chapter. The depiction of this kind of rule in [4] [9] is so brief that they could not emphasize the sensitive relation between fuzzy association rules and fuzzy logic theory. The thesis also explains why we choose *min* function and *algebraic multiplication* for T-norm operator in formula (3.6). In addition, this section restates the algorithm for mining fuzzy association rules in [4] [9] based on Apriori with just slight customizations. Finnaly, this

chapter offers a transformation method for converting fuzzy association rule into quantitative ones.

Chapter four suggests a novel parallel algorithm for mining fuzzy association rules. In this algorithm, processors will largely reduce the amount of information need to be communicated during processing time. The algorithm is considered to be "ideal" thanks to its wise strategy in deliver the original set of fuzzy attributes for processors. This division method is both balanced and intelligent in that partitions after dividing are equal and each processor can operate upon its partition independently. However, the algorithm contains some drawbacks. First, it is only successfully applied for mining fuzzy or quantitative association rules. Second, it only works well on shared-nothing parallel systems.

During the time I write this thesis and previous time, I have tried my best to research and consult many other reference documents. However, the thesis cannot avoid several mistakes and shortcomings. I would like to appreciate any comment and suggestion in both technical and editorial problems from all readers.

5.2 Future research

Association rule mining is attractive to many researchers because of its widespread applications in various areas as well as it potentially contains different research trends. In this dissertation, I only focus on *fuzzy association rule* and parallel algorithm for mining this kind of rule. In the next time, the following research directions will be taken into consideration:

- Mining fuzzy association rules with weighted attributes. The major goal of this kind of rule is to measure the "contribution level" of each attribute to the overall rule. For example, as we mining association rules on data related to heart disease, the information of attributes such as *blood pressure*, *blood sugar*, *cholesterol* is much more critical than that of *body weight* and *age*. Thus, we attach higher weights to *blood pressure*, *blood sugar*, *cholesterol* comparing to those of *body weight* and *age*.
- The parallel algorithm we recommend in chapter four works only on shared-nothing systems. In the near future, we would like to concentrate on designing parallel algorithms for other architectures such as SMP, etc.

- Although the issue of association rules mining is highly independent to what kind of data to be mined. However, we would like to focus on a certain database to improve the proposed algorithms and to tune related parameters.

Reference

Vietnamese documents:

[1]. [PDD99] Phan Dinh Dieu. Logic in knowledge systems. Faculty of Technology, Vietnam National University, Hanoi – 1999.

[2]. [DMT03] Dinh Manh Tuong. Artificial Intelligence. Faculty of Technology, Vietnam National University, Hanoi – 2003.

English documents:

[3]. [AR95] Alan Rea. Data Mining – An Introduction. The Parallel Computer Centre, Nor of The Queen's University of Belfast.

[4]. [AG00] Attila Gyenesei. A Fuzzy Approach for Mining Quantitative Association Rules. Turku Centre for Computer Science, TUCS Technical Reports, No 336, March 2000.

[5]. [AM95] Andreas Mueller. Fast Sequential and Parallel Algorithms for Association Rule Mining: A Comparison. Department of Computer Science, University of Maryland-College Park, MD 20742.

[6]. [LHM99] Bing Liu, Wynne Hsu, and Yiming Ma. Mining Association Rules with Multiple Minimum Supports. In *ACM SIGKDD* International Conference on KDD & Data Mining (KDD-99), August 15-18, 1999, San Diego, CA, USA.

[7]. [KV01] Boris Kovalerchuk and Evgenii Vityaev. Data Mining in Finance – Advances in Relational and Hybrid Methods. Kluwer Academic Publishers, Boston – Dordrecht - London. 2001.

[8]. [MHT02] Bui Quang Minh, Phan Xuan Hieu, Ha Quang Thuy. Some Parallel Computing Experiments with PC-Cluster. In *Proc. of Conference on IT of Faculty of Technology, VNUH*. Hanoi 2002.

[9]. [KFW98] Chan Man Kuok, Ada Fu, and Man Hon Wong. Mining Fuzzy Association Rules in Databases. Department of Computer Science and Engineering, The Chinese University of Hong Kong, Shatin, New Territories, Hong Kong.

[10]. [THH02] Do Van Thanh, Pham Tho Hoan, and Phan Xuan Hieu. Mining Association Rules with different supports. In *Proc. of the National Conference on Information Technology*, Nhatrang, Vietnam, May 2002.

[11]. [BCJ01] Doug Burdick, Manuel Calimlim, and Johannes Gehrke. MAFIA: A Maximal Frequent Itemset Algorithm for Transactional Databases. Department of Computer Science, Cornell University.

[12]. [HKK97] Eui-Hong (Sam) Han, George Karypis, and Vipin Kumar. Scalable Parallel Data Mining for Association Rules. Department of Computer Science, University of Minnesota, 4-192 EECS Building, 200 Union St. SE, Minneapolis, MN 55455, USA.

[13]. [PHM01] Jian Pei, Jiawei Han, and Runying Mao. CLOSET: An Efficient Algorithm for Mining Frequent Closed Itemsets. Intelligent Database Systems Research Lab, School of Computing Science, Simon Fraser University, Burnaby, B.C., Canada.

[14]. [HK02] Jiawei Han and Micheline Kamber. Data Mining: Concepts and Techniques. University of Illinois, Morgan Kaufmann Publishers 2002.

[15]. [HF95] Jiawei Han and Yongjian Fu. Discovery of Multiple-Level Association Rules from Large Databases. In *Proc. of the 21st International Conference on Very Large Dadabases,* Zurich, Switzerland, Sep 1995.

[16]. [LZDRS99] Jinyan Li, Xiuzhen Zhang, Guozhu Dong, Kotagiri Ramamohanarao, and Qun Sun. Efficient Mining of High Confidence Association Rules without Support Thresholds. Department of CSSE, The University of Melbourne, Parkville, Vic, 3052, Australia.

[17]. [HG00] Jochen Hipp, Ulrich Guntzer, and Gholamreza Nakhaeizadeh. Algorithms for Association Rule Mining – A General Survey and Comparison. *ACM SIGKDD*, July 2000, Volume 2, Issue 1 – page 58 – 64.

[18]. [PCY95] Jong Soo Park, Ming-Syan Chen, and Philip S. Yu. Efficient Parallel Data Mining for Association Rules. In *Fourth International Conference on Information and Knowledge Management,* Baltimore, Maryland, Nov 1995.

[19]. [PYC98] Jong Soon Park (Sungshin Women's Univ, Seoul, Korea), Philip S. Yu (IBM T.J. Watson Res. Ctr.), and Ming-Syan Chen (National Taiwan Univ., Taipei, Taiwan). Mining Association Rules with Adjustable Accuracy.

[20]. [MTV94] Heikki Mannila, Hannu Toivonen, and A. Inkeri Verkamo. Efficient Algorithms for Discovering Association Rules. In *KDD-1994: AAAI Workshop on Knowledge Discovery in Databases,* pages 181-192, Seattle, Washington, July 1994.

[21]. [LAZ65] L. A. Zadeh. Fuzzy sets. Informat. Control, 338-353, 1965.

[22]. [KMRTV94] M. Klemettinen, H. Mannila, P. Ronkainen, H. Toivonen, and A.I. Verkamo. Finding Interesting Rules from Large Sets of Discovered Association Rules. In *Proc. 3rd International Conference on Information and Knowledge Management,* pages 401-408, Gaithersburg, Maryland, November 1994.

[23]. [MM00] Manoel Mendonca. Mining Software Engineering Data: A Survey. University of Maryland, Department of Computer Science, A. V. Williams Building #3225 College Park, MD 20742. 2000.

[24]. [ZH99] Mohammed J. Zaki and Ching-Jui Hsiao. CHARM: An Efficient Algorithm for Closed Association Rules Mining. RPI Technical Report 99-10, 1999.

[25]. [ZO98] Mohammed J. Zaki and Mitsunori Ogihara. Theoretical Foundations of Association Rules. In *3rd ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery,* June 1998.

[26]. [ZPO01] Mohammed J. Zaki, Srinivasan Parthasarathy, and Mitsunori Ogihara. Parallel Data Mining for Association Rules on Shared-Memory Systems. In *Knowledge and Information Systems,* Vol. 3, Number 1, pages 1-29 February 2001.

[27]. [MPIS95] *MPI: A Message-Passing Interface Standard*, Message Passing Interface Forum. June 12, 1995.

[28]. [EMPI97] *MPI-2: Extensions to the Message-Passing Interface*, Message Passing Interface Forum, July 18, 1997

[29]. [JDMPI97] *MPI-2 Journal of Development*, Message Passing Interface Forum, July 18, 1997.

[30]. [PBTL99] Nicolas Pasquier, Yves Bastide, Rafik Taouil, and Lotfi Lakhal. Discovering Frequent Closed Itemsets for Association Rules. Laboratoire d'Informatique, Université Blaise Pascal – Clermont-Ferrand II, Complexe Scientifique des Cézeaux.

[31]. [ZHL98] Osmar R. Zaiane, Mohammad El-Hajj, and Paul Lu. Fast Parallel Association Rule Mining Without Candidacy Generation. University of Alberta, Edmonton, Alberta, Canada.

[32]. [DP01] Qin Ding and William Perrizo. Using Active Networks in Parallel Mining of Association Rules. Computer Science Department, North Dakota State University, Fargo ND 58105-5164.

[33]. [AY98] R. Agrawal and P. Yu. Online Generation of Association Rules. In *IEEE International Conference on Data Mining,* February 1998.

[34]. [AS96] Rakesh Agrawal and John Shafer. Parallel mining of association rules: Design, implementation and experience. Research Report RJ 10004, IBM Almaden Research Center, San Jose, California, February 1996.

[35]. [AS94] Rakesh Agrawal and Ramakrishnan Srikant. Fast Algorithms for Mining Association Rules. In *Proc. of the 20th International Conference on Very Large Databases,* Santiago, Chile, Sep 1994.

[36]. [AIS93] Rakesh Agrawal, Tomasz Imielinski, and Arun Swami. Mining association rules between sets of items in large databases. In *Proc. of theACM SIGMOD Conference on Management of Data,* pages 207-216, Washington, D.C., May 1993.

[37]. [SA95] Ramakrishnan Srikant and Rekesh Agrawal. Mining Generalized Association Rules. In *Proc. of the 21st International Conference on Very Large Databases,* Zurich, Switzerland. Sep 1995.

[38]. [SA96] Ramakrishnan Srikant and Rakesh Agrawal. Mining Quantitative Association Rules in Large Relational Tables. IBM Almaden Research Center, San Jose, CA 95120.

[39]. [MY98] R. J. Miller and Y. Yang. Association Rules over Interval Data. Department of Computer & Information Science, Ohio State University, USA.

[40]. [PMPIP] *RS/6000 SP: Practical MPI Programming*, Yukiya Aoyama & Jun Nakano, International Technical Support Organization, www.redbooks.ibm.com

[41]. [MS00] T. Murai and Y. Sato. Association Rules from a Point of View of Modal Logic and Rough Sets. In proceeding of the forth Asian Fuzzy Symposium, May 31, June 3, 2000, Tsukuba, Japan, pp, 427-432.

[42]. [HHMT02] Tran Vu Ha, Phan Xuan Hieu, Bui Quang Minh, and Ha Quang Thuy. A Model for Parallel Association Rules Mining from The Point of View of Rough Set. In *Proc. of International Conference on East-Asian Language Processing and Internet Information Technology*, Hanoi, Vietnam, January 2002.

[43]. [FSSU96] Usama M. Fayyad, Gregory Piatetsky-Shapiro, Padhraic Smyth, and Ramasamy Uthurusamy. Advances in Knowledge Discovery and Data Mining. AAAI Press / The MIT Press 1996.

[44]. [WYY01] Wei Wang, Jiong Yang, and Philip S. Yu. Efficient Mining of Weighted Association Rules (WAR). IBM Watson Research Center.

[45]. [WMPPP] *Writing Message-Passing Parallel Programs with MPI*, Neil MacDonald, Elspeth Minty, Tim Harding, Simon Brown, Edinburgh Parallel Computing Centre, The University of Edinburgh.

[46]. [ZKM01] Zijian Zheng, Ron Kohavi, and Llew Mason. Real World Performance of Association Rule Algorithms. Blue Martini Software, 2600 Campus Drive, San Mateo, CA 94403, USA.

[47]. [ZHJ91] Zimmermann H. J. Fuzzy Set Theory and Its Applications. Kluwer Academic Publishers, 1991.