SQL IMPLEMENTATION OF VALUE REDUCTION WITH MULTISET DECISION TABLES

A Thesis

Presented to

The Graduate Faculty of The University of Akron

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

Chen Chen

May, 2014

SQL IMPLEMENTATION OF VALUE REDUCTION

WITH MULTISET DECISION TABLES

Chen Chen

Thesis

Approved:	Accepted:
Advisor Dr. Chien-Chung Chan	Dean of the College Dr. Chand Midha
Committee Member Dr. Yingcai Xiao	Dean of the Graduate School Dr. George R. Newkome
Committee Member Dr. Zhong-Hui Duan	Date
Department Chair Dr. Yingcai Xiao	

ABSTRACT

Data reduction is an important contribution of rough set theory in data analysis, data mining and machine learning. Most recent researches have focused on attribute reductions. However, another main part of reduction, value reduction, has barely been paid attention to for a long period of time. Value reduction can induce decision rules with logical equivalent minimal length.

Some research ideas such as high frequency value reduction and association rules have been applied to value reduction to gain better generating performance. This research introduced a new value reduction algorithm combining rough set theory with association rules to generate decision rules from examples in format of Multiset Decision Tables (MDT). Testing with the University of California Irvine (UCI) machine learning repository, the comparison was performed between this algorithm and high frequency value reduction algorithm. The result indicated the fitness of the new algorithm to process large data and validation to improve.

ACKNOWLEDGEMENTS

I would like to thank my revered advisor, Professor Dr. Chien-Chung Chan for his valuable advice and guidance throughout all phases of this work. Without his patience and help, it would have never been completed. I also thank my parents and my sister for their unconditional support.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER I

INTRODUCTION

1.1 Thesis Statement

In this chapter, the research is described briefly. It describes the background of the research, the problem statement, the motivation and the scope of the research. It is not only the preliminary description of the research but also the functional summary of the entire project. At the end of this chapter, an overall organization of the following chapters is listed to give readers the entire picture of the thesis.

1.2 Motivation

With the explosive growing data creation, digital data stored in databases has increased exponentially. Although, the rapid advancement of micro-electrics also hugely decreases the budgetary pressure in data storage, it is still a challenging problem to find out valuable knowledge in flooding information. Therefore the main goal of data mining involves in effort of determining valuable information efficiently from tremendous data repository.

Rough set theory (RST) provides an elegant and powerful method to extract rules from decision tables. Researchers have invested resources to investigate the relationships of attributes within large data set with rough set theory. Reduct is a main aspect of all these researches. Most of these researches focus on attributes reduction. However, value reduction, although an important aspect of reduct, was not paid so much attention in past researches.

In this thesis, we studied and implemented an existing value reduction algorithm and proposed a new algorithm. Meanwhile, we also compared the existing and the new algorithm to find out the advantages and disadvantages. Our motivation is to design a reduction algorithm combining with rough set theory and association rules to extract important rules from large scale databases with less time and space.

1.3 Scope of the work

In this thesis, research focuses on a value reduction algorithm combining rough set theory and association rule mining together. A new algorithm is proposed to handle value reduction in large size datasets.

The algorithm will be tested on the dataset from UCI machine learning repository KDDCUP 1999. For the convenience of research, we applied normalization and discretization to the data set before the value reduction step began (which will be introduced in Chapter 3).

At the comparison step, the algorithm was tested in 10 different sizes of the dataset from small to large. On each scale of size, we tested the existing algorithm, the newly proposed algorithm and the decision tree algorithm to evaluate the efficiency of value reduction with regard to the algorithm.

1.4 Thesis Organization

This thesis is organized as followings:

Chapter 2 is the review of the preliminary concept of rough set theory and association rule mining. This part began with a brief introduction of Rough Set Theory. Then the discussion of rough reducts was given which consists of the focus of the research. Next, we continued the introduction of multiset decision tables (MDT) and association rule mining. At last, the attribute reduction, value reduction techniques and method were reviewed in general.

Chapter 3 presents the methodology of the research. It starts from the data cleaning as a pre-processing step, then the implementation of generating MDT to reduce the quantity of raw data. Then it follows with the implementation of existing value reduction algorithm and the implementation of proposed value reduction algorithm combined with association rule mining technique.

Chapter 4 includes the fundamental theory for the proposed algorithm. It contains two major parts that illustrate the existing value reduction algorithm and proposed value reduction algorithm techniques respectively.

The implementation and test of two algorithms are presented in Chapter 5. The second part of Chapter 5 describes analysis of the experiment results of both existing and proposed value reduction algorithm. Chapter 6 concludes the thesis and future work.

CHAPTER II

FUNDAMENTAL CONCEPTS

2.1 KDD and Data mining

Definition 2.1: knowledge discovery in Database (KDD)

It is the process of identifying valid, novel, potentially useful, and ultimately understandable structure in data [1].

Definition 2.2: Data Mining

It is a step in the KDD process that, under the acceptable computational efficiency limitation, enumerates structures (patterns or models) over the data [1]. Data Mining refers to a particular step in the process of KDD. The additional steps in the KDD process, such as data preparation, data selection, data cleaning, incorporating appropriate prior knowledge, and proper interpretation of the results of mining, are essential to ensure that useful knowledge is derived from the data.

2.2 Rough Set Theory

Rough set theory was introduced by Zdzislaw Pawlak in 1980's [2, 3, 4, 5]. At the beginning, it was seen as an extension of the classical set theory to represent the incomplete knowledge. Rough set theory can be used to handle sets that cannot be precisely described. As part of the fundamental research for information system, rough set theory has been widely applied as important tool for data analysis.

The methodology of rough set mainly focuses on the analysis of the data acquired from the experience which is imprecise, uncertain or incomplete. It provides powerful tools to resolve imprecise and ambiguous data in data mining and data exploration.

Recently, these tools are widely used in the industrial data mining processes such as marketing, disease diagnosis, IT security analysis and financial analysis. These rough set models provide useful experience to extend the original models and help solve some limitations in performance.

2.2.1 Information System

In Rough Set Theory, an information system is usually defined as a pair $IS = (U, A, V, f)$, where U = {u1, u2, ..., un} is a non-empty finite set called the universe and A is a non-empty finite set of attributes, i.e., a: U →Va for a ∈ A, where Va is called the domain of a. Function of information system can be defined as $f : U \times A \to V$ that $f(x, q) \in V_q$ for every $q \in A$ and $x \in U$.

2.2.2 Decision Table

Normally, decision table is considered as a special case of information system. It is used to represent the information system to specify what conditions result in the decisions.

In decision table S = (U, C, D, {Va} a∈C ∪ D, f), attributes in C are called condition attributes, D is a designated attribute called the decision attribute and C ∩ D = ϕ, Va is the domain of attribute a. f is a function: U × (C ∪ D) → $V = \bigcup_{a \in C \cup D} V_a$ which maps each pair of tuple and attribute to an attribute value.

We have an example of how decision table looks like in Table 2.1. This table shows a data set about cars [6]:

Table 2.1: An example of decision table

	Maker	cyl	door	displace	compress	power	trans	weight	Mileage
1	USA	6	2	Medium	High	High	Auto	Medium	Medium
2	USA	6	4	Medium	Medium	Medium	Manual	Medium	Medium
3	USA	4	2	Small	High	Medium	Auto	Medium	Medium
4	USA	4	2	Medium	Medium	Medium	Manual	Medium	Medium
5	USA	4	2	Medium	Medium	High	Manual	Medium	Medium
6	USA	6	4	Medium	Medium	High	Auto	Medium	Medium
7	USA	4	2	Medium	Medium	High	Auto	Medium	Medium
8	USA	4	2	Medium	High	High	Manual	Light	High
9	Japan	4	2	Small	High	Low	Manual	Light	High
10	Japan	4	2	Medium	Medium	Medium	Manual	Medium	High
11	Japan	4	2	Small	High	High	Manual	Medium	High
12	Japan	4	2	Small	Medium	Low	Manual	Medium	High
13	Japan	4	2	Small	High	Medium	Manual	Medium	High
14	USA	4	2	Small	High	Medium	Manual	Medium	High
15	USA	6	2	Medium	High	High	Auto	Medium	Medium
16	Japan	4	2	Small	High	High	Manual	Medium	High

Table 2.1 shows a decision table about cars with 9 attributes totally. The mileage of sedan is determined by the maker, the number of cylinders, the number of doors, the displacement, the compression, the power, the transmission, and the weight of the car.

For instance, in the first row of the Table 2.1, this car is made by USA, 6 cylinders, 2 doors, medium displacement, high compression, high power, auto transmissions, and medium weight. It's the car with medium mileage.

Thus, in this decision table S, condition attributes set C is the set {maker, cylinder, door, displace, compress, power, transmission, weight} and the attribute set D = {Mileage} is the decision attribute.

2.2.3 Multiset Decision Table(MDT)

Table 2.2: Example of Decision Table

No.	C1	C2	C3	D
1	1	2	3	2
2	1	3	3	2
3	1	2	1	1
4	1	1	1	1
5	3	3	3	3
6	1	2	2	2
7	1	2	2	1
8	2	3	3	3
9	1	2	2	1
10	1	2	2	2
11	2	3	3	2
12	1	2	2	2
13	1	2	2	2
14	1	2	2	1
15	1	3	3	2
16	2	3	3	3
17	3	4	4	3

A multi-set decision table is the concept that derived from an information multisystem. In [7], the basic idea of MDT was introduced to display distinct decision values and it supports weight in separate columns instead of rows. Table 2.3 shows an example of information multisystem generated from Table 2.2. Each value in column wt indicates how many times this row appears in the original decision table.

Table 2.3: an information multisystem

C1	C2	C3	D	wt
1	1	1	1	1
1	2	1	1	1
1	2	2	1	3

C1	C2	C3	D	wt
1	2	2	2	4
1	2	3	2	1
1	3	3	2	2
2	3	3	2	1
2	3	3	3	2
3	3	3	3	1
3	4	4	3	1

The idea of multi-set decision tables (MDT) was first introduced by Chan [8] which gave the definition of Multiset decision tables as below:

Let S = (Q = C $\cup$ D, V, $\widetilde{Q}$) be an information multisystem, where C are condition attributes and D is a decision attribute. A multi-set decision table is an ordered pair A = ($\widetilde{C}$, CD), where $\widetilde{C}$ is a projection of $\widetilde{Q}$ onto C and CD is a multi-partition on $\widetilde{D}$ generated by C in A. We will call $\widetilde{C}$ the LHS (Left Hand Side) and CD the RHS (Right Hand Side). Blocks in CD are subsets of decision values, which are represented as a Boolean vector and the number of occurrences is represented by an integer vector.

From the example in Table 2.4, MDT is composed of two parts, LHS (Left Hand Side) and RHS (Right Hand Side). In LHS, all the columns denote the condition attributes and the occurrences in decision table. RHS then store the decision-value columns and corresponding weight distributions for decision-value.

Table 2.4: MDT example

No	C1	C2	C3	wt	D1	D2	D3	W1	W2	W3
1	1	1	1	1	1	0	0	1	0	0
2	1	2	1	1	1	0	0	1	0	0
3	1	2	2	7	1	1	0	3	4	0
4	1	2	3	1	0	1	0	0	1	0
5	1	3	3	2	0	1	0	0	2	0
6	2	3	3	3	0	1	1	0	1	2

No	C1	C2	C3	wt	D1	D2	D3	W1	W2	W3
7	3	3	3	1	0	0	1	0	0	1
8	3	4	4	1	0	0	1	0	0	1

There are three distinct values of decision attribute D in Table 2.3 which are 1, 2 and 3. Therefore, in RHS three boolean columns D1, D2, and D3 are added. The weight column means the number of occurrences of rows over the LHS columns. The weight distribution of decision values D1, D2, D3 corresponding to LHS columns are W1, W2, W3. Therefore, weight in LHS is the sum of w1, w2 and w3.

2.2.4 Rough Sets Reducts and Value Reduction

Reduct is an important concept in rough sets theory. From the original data set, a reduct contains a minimal subset of the attributes which have the equivalent discernible capability as the original information system [9].

There are three main aspects of reduction in rough set theory: attribute reduction, row reduction, and value reduction.

Row reduction only merges duplicate rows. It can be easily conducted by a group by operation in SQL.

This attribute reduction is usually used in a process to remove superfluous attributes, in column direction, and contain the same classification with the original information system.

Value reduction simplifies decision rules. It removes irrelevant data from decision rules which brings equivalent effect compared to the original rules. Since the concept of value reduction was introduced, different algorithms about value reduction have been developed for implementation. Based on several typical value reduction algorithms, the proposed algorithm try to provide privileges in different aspects.

2.3 Association Rules Mining

In data mining, the association rule algorithm [10] is known as one method to discover correlation between variables in huge dataset.

The typical example of association rules mining is to discover association from transaction datasets. In this algorithm, frequent itemset which contains items that have support more than minimum support is generated first. Based on these frequent itemsets, the algorithm discovers the association rule which matches the minimum confidence requirements. In this case, support measures how often this rule emerges in the transaction set.

Recently, many contributions have been reported to generate frequent itemsets and rules with efficient methods [11,12,13]. Support is often used to evaluate rough set based rules. Generally, a rule is considered qualified when this rule has support more than minimum support. With management of minimum support, people can control the granularity of rule generation, and it is possible to get the whole rule set by setting minimum support equal to 1.

CHAPTER III

RESEARCH METHODOLOG

3.1 Introduction

In this chapter, the methodology of this research and path on how to achieve the project are discussed. Knowledge Discovery in Databases (KDD) was chosen as the approach to test the targeted algorithm. By following the direction of KDD, the hidden pattern was able to be identified and extracted from the real-world databases. Also, this hidden pattern can be translated into the understandable knowledge which may bring great benefits for specific purpose. In this case, a test dataset with more than 4 million instances was configured as very large database in the system. To handle such large dataset, it is impossible to transform the data into valid, novel knowledge with traditional method (e.g. manual analysis). Therefore, KDD was chosen to generate useful knowledge in a much more efficient manner rather than using the traditional analysis method.

3.2 Knowledge Discovery in Databases

As mentioned before, Knowledge Discovery in Databases (KDD) provides methodology for extracting useful knowledge from data. As shown in Figure 3.1

below, there are 5 phases in KDD, which are selection, preprocessing, transformation, data mining, and interpretation and evaluation.

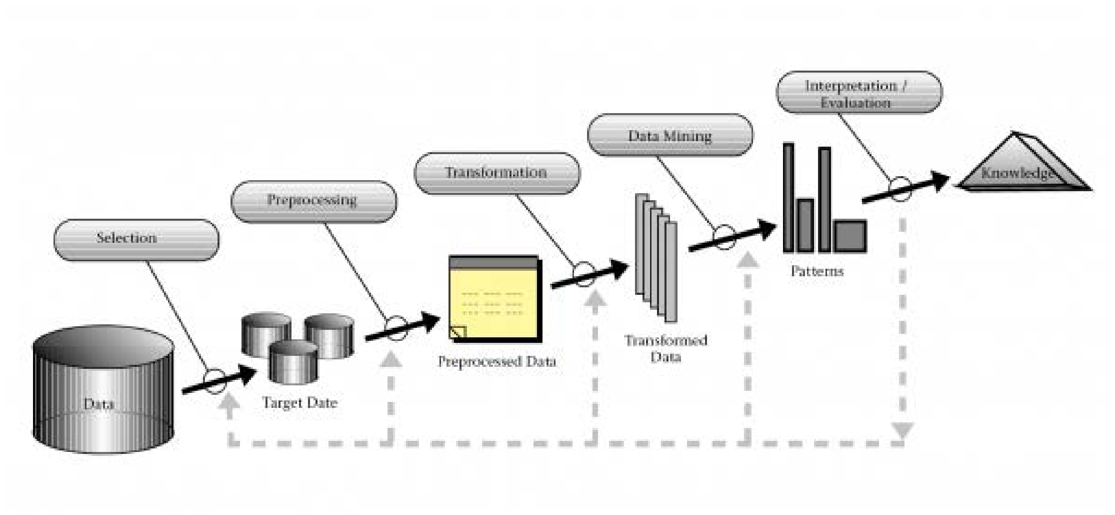


Figure 3.1: An Overview of the Steps of the KDD Process [1]

We will introduce the concept and the detailed information related to this project in the following subsections.

3.2.1 Dataset Selection

Before the data mining can be performed, one target data set needs to be selected. This data set must be large enough to contain knowledge while remaining concise enough to be extracted in an efficient manner.

In this case, we choose Intrusion Detection System (IDS) Evaluation dataset as the one we use throughout the project. Intrusion Detection System (IDS) Evaluation dataset was created by Lincoln Lab for KDD CUP 1999 and retrieved from the open-source website UCI Machine Studying Repository. This dataset is composed of more than 4 million instances, 42 attributes mixed with continuous and discrete values.

The dataset simulates a network intrusion detector which can distinguish all connection into “malicious” connections, called as intrusions or attacks, and “not malicious” connections. In more than 4 million instances, IDS simulates a wide variety of intrusions and the standard data to be audited in a military network environment.

In the table listed below it lists the explanation of each field in the dataset.

Table 3.1: Attribute Descriptions in IDS

	Feature Name	Description of Features	Type
1	duration	length (number of seconds) of the connection	continuous
2	protocol_type	type of the protocol, e.g. tcp, udp, etc.	symbolic
3	service	network service on the destination, e.g., http, telnet, etc.	symbolic
4	flag	normal or error status of the connection	symbolic
5	src_bytes	number of data bytes from source to destination	continuous
6	dst_bytes	number of data bytes from destination to source	continuous
7	land	1 if connection is from/to the same host/port; 0 otherwise	discrete
8	wrong_fragment	number of ``wrong'' fragments	continuous
9	urgent	number of urgent packets	continuous
10	hot	number of ``hot'' indicators	continuous
11	num_failed_logins	number of failed login attempts	continuous
12	logged_in	1 if successfully logged in; 0 otherwise	discrete
13	num_compromised	number of ``compromised'' conditions	continuous
14	root_shell	1 if root shell is obtained; 0 otherwise	discrete
15	su_attempted	1 if ``su root'' command attempted; 0 otherwise	discrete
16	num_root	number of ``root'' accesses	continuous
17	num_file_creations	number of file creation operations	continuous
18	num_shells	number of shell prompts	continuous
19	num_access_files	number of operations on access control files	continuous
20	num_outbound_cmds	number of outbound commands in an ftp session	continuous
21	is_hot_login	1 if the login belongs to the ``hot'' list; 0 otherwise	discrete
22	is_guest_login	1 if the login is a ``guest''login; 0 otherwise	discrete
23	count	number of connections to the same host as the current connection in the past two seconds	continuous
24	srv_count	number of connections to the same service as the current connection in the past two seconds	continuous
25	serror_rate	% of connections that have ``SYN'' errors	continuous
26	srv_serror_rate	% of connections that have ``SYN'' errors	continuous

	Feature Name	Description of Features	Type
27	rerror_rate	% of connections that have ``REJ" errors	continuous
28	srv_rerror_rate	% of connections that have ``REJ" errors	continuous
29	same_srv_rate	% of connections to the same service	continuous
30	diff_srv_rate	% of connections to different services	continuous
31	srv_diff_host_rate	% of connections to different hosts	continuous
32	dst_host_count		continuous.
33	dst_host_srv_count:		continuous.
34	dst_host_same_srv_rate:		continuous.
35	dst_host_diff_srv_rate:		continuous.
36	dst_host_same_src_port_rate:		continuous.
37	dst_host_srv_diff_host_rate:		continuous.
38	dst_host_serror_rate:		continuous.
39	dst_host_srv_serror_rate:		continuous.
40	dst_host_rerror_rate:		continuous.
41	dst_host_srv_rerror_rate:		continuous.
42	Decision	Normal or attack connection	symbolic

3.2.2 Preprocessing

After selecting the target dataset, a preprocessing step needs to be performed before the dataset can be used in the analysis.

Normally, the process of collecting data is controlled loosely. It causes several types of defective data, such as out-of-range value (e.g., Age: −50), impossible data combination, incomplete, inconsistent and noisy data. It may lead to imprecise results if these defective data are not carefully screened. Hence, data preprocessing, the process of data correcting, cleaning and quality improving, are implemented to obtain correct and clean data without noise.

The next stage is normalization. It’s the process of transforming the data into predefined group; in this research only the nominal data are normalized.

The normalization for the data is given in Table 3.5 and the sample of normalized IDS dataset are shown in Table 3.2.

Table 3.2: Example of Attributes Normalization

Features Name	Normalization Format

protocol_type	Protocol_type='1' where Protocol_type='icmp'
	Protocol_type='2' where Protocol_type='tcp'
	Protocol_type='3' where Protocol_type='udp'

service	Service='1' where Service='aol'
	Service='2' where Service='auth'
	Service='3' where Service='bgp'
	.
	.
	.
	Service='70' where Service='z39_50'

flag	flag='1' where flag='OTH'
	flag='2' where flag='REJ'
	flag='3' where flag='RSTO'
	flag='4' where flag='RSTOS0'
	flag='5' where flag='RSTR'
	flag='6' where flag='S0'
	flag='7' where flag='S1'
	flag='8' where flag='S2'
	flag='9' where flag='S3'
	flag='10' where flag='SF'
	flag='11' where flag='SH'

decision	Decision = '1' where Decision='back.'
	Decision = '2' where Decision='buffer_overfl
	Decision = '3' where Decision='ftp_write.'
	Decision = '4' where Decision='guess_passwd.
	Decision = '5' where Decision='imap.'
	Decision = '6' where Decision='ipsweep.'
	Decision = '7' where Decision='land.'
	Decision = '8' where Decision='loadmodule.'
	Decision = '9' where Decision='multihop.'
	Decision = '10' where Decision='neptune.'
	Decision = '11' where Decision='nmap.'
	Decision = '12' where Decision='normal.'
	Decision = '13' where Decision='perl.'
	Decision = '14' where Decision='phf.'
	Decision = '15' where Decision='pod.'
	Decision = '16' where Decision='portsweep.'
	Decision = '17' where Decision='rootkit.'

Features Name	Normalization Format

	Decision = '18' where Decision='satan.'
	Decision = '19' where Decision='smurf.'
	Decision = '20' where Decision='spy.'
	Decision = '21' where Decision='teardrop.'
	Decision = '22' where Decision='warezclient.
	Decision = '23' where Decision='warezmaster.

3.2.3 Transforming

Data transformation is the phase of transforming the values of the dataset from the source data format to the target data format of destination data system.

In this step, data should be converted to meet two main goals. First, the dataset needs to be changed into the format that understandable by the data analysis software. Second, we need to implement certain optimization to get a more compact, efficient data format for the analysis step.

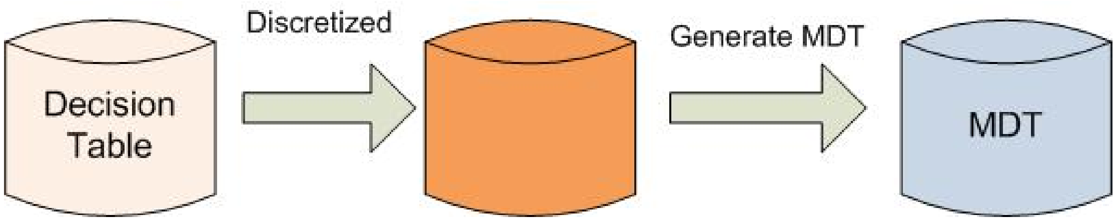


Figure 3.2: Transformation Process

Therefore, at first, the dataset needs to be transformed into the discretized format by applying algorithm of discretization. Normally, discretization is a process that transfers the continuous models and equations into discrete counterparts, which makes the unique value minimized. For example, the continuous attributes will cause many meaningless rules. In discretization step, it will be divided into a few intervals which express as much information as possible to the rules. Each interval generates one kind of mapping relation with a discrete symbol, like nominal, categorical and symbolic. Most of the analysis tools provide discretization function, SQL Server 2005 Analysis Service provides several options for the discretization; for instance, Clusters and Equal areas algorithm. In this research, SQL Server Analysis Services determines discretization method by choosing automatic option which brings us a compromise between efficiency and accuracy. With the discretization step, the continuous attributes become meaningful and efficient for the upcoming analysis step.

The next phase followed is generating MDT whose records are much less than the original dataset [8]. The detailed concept has been described in session 2.2.3. After the MDT is generated, our dataset should be clean and correct, and the format is simplified and ready for analysis.

3.2.4 Reduct

Value reduction is a process that removes non-relevant attributes from decision rules. The central notions of this research are to perform the value reduction without finding attribute reducts.

At this stage, two value reduction algorithms will be developed to compute and simplify the decision rules. High Frequent Value Reduct algorithm (HFVR) [14] and rough set–association rule mining value reduct algorithm (RSAVR). Basically, they are all based upon the high frequent item sets appearing in the dataset.

The main content and implementation of these two algorithms will be described in Chapter 4 with detailed explanation of the algorithms.

3.2.5 Comparison and Evaluation

The final step of this thesis is comparison, which includes the differences and the performance of the two algorithms.

The comparison will demonstrate the efficiency of the algorithms, which should be an indication of the success of the research, and may be of great interest to other researchers.

3.3 Summary

In the progress of project, KDD is chosen as the methodology for the research. In our methodology, five phases including selection, preprocessing, transformation, reduction and comparison are chosen for implementation in order to demonstrate the idea of the project.

CHAPTER IV

HIGH FREQUENT VALUE REDUCT ALGORITHM (HFVR)

This chapter focuses upon the presentation of an existing classic algorithm of value reduction, HFVR (high frequent value reduct). This algorithm proposed an approach of inducing decision rules by allocating high frequent value reduct directly without finding any attribute reduct [14]. This part summarizes the main idea and distinction in [14] and gives the readers both the overview and the detailed information for this algorithm.

4.1 HFVR Algorithm Description

Before proceeding with the description of the algorithm, it should be indicated that only update of relational database are taken into consideration. Therefore, the implementation of creating MDT for the decision table is pre-executed before starting the process of this algorithm and will not be discussed in the following sections.

After the MDT is prepared successfully, the proposed algorithm can be summarized in the following steps:

Step 1: Create the MDT T for decision table;

Step 2: Set K as 1, this is the subset size of condition attributes taken to detect candidate rule, construct the attributes set Ck;

Step 3: For each C in Ck, find a subset TA of T with all the values in C's value;

Step 4: Delete the rules from TA if rule support is less than or equal to the threshold of support. Remove inconsistent rules in TA;

Step 5: Check if rules in TA is covered by rule set R, if true, remove covered rules from TA;

Step 6: Put the tuple remained in TA into rule set R;

Step 7: Increase K to K+1, repeat Steps 3 to 6 until Ck equals to set C;

Step 8: End.

Starting from step 2, a subset is created which contains all the elements with size of k from the power set of condition attributes set. For instance, decision table (Table 2.1) has three condition attributes C1, C2 and C3; thus condition attributes set should be {C1, C2, C3}. The parameter K indicates size should start from 1 then increase to 3 till the end. When K equals to 1, Ck will be {C1, C2, C3}; then, increase k to 2, Ck will be changed to {(C1, C2), (C1, C3), (C2, C3)} before the execution of step 4 to 6.

From the description above, it is clear that the power set needs to be generated over the condition attributes set C except for the empty set; therefore, at least it needs to scan 2^n-1 times over the database to find the value reducts.

The overview of the algorithm is shown in the following [14].

Algorithm: Finding all decision rules in a decision table by value reduction
Input: A decision table *T* in a relational table with condition attribute set *C*, and a decision attribute *d;* a minimum support threshold *s*
Output: *RB,* a set of decision rules
High Frequent Value Reduct in Very Large Databases
Procedure:

```
1.  RB ←empty
2.      For k=1 to |C| Do
3.          RBk ←empty
4.          For each subset of C, A of size k, Do
5.              TA ← create a subset from T with all columns in A
6.              Remove all inconsistent and insufficient support tuples from TA
7.              For each remaining tuple r in TA Do
8.                  If r is not covered by R Then RBk ← RBk ∪{r}
9.              If RBk = empty Then Return RB
10.             Else R ←□R ∪RBk
11. Return R
End
```

Figure 4.1: A high-level description of existed algorithm

4.1.1 Handle Inconsistent and Insufficient Support

Before describing handling inconsistent and insufficient supported rules, an important concept, the inconsistent tuples, should be defined as below:

Definition 4.1. U is consistent if no contradictory pair of tuples exist in U, that is, random t1, t2 $\in$ *U, if t1[D] = t2[D], then t1[C] = t2[C].*

Definition 4.2. In a decision table, C and D are condition and decision attributes respectively. The rule C =>D has support s in the transaction set T if s of transactions in T contain C $\cup$ *D.*

Inconsistency means there are conflicts existing in rule objects. Although these rules were characterized by the same values of all condition attributes, they induce different decision attribute values. Such object should be removed from test data set.

Support is an important concept derived from association rules mining. With management of the minimum support, people can control the granularity of rule generation. In the relational database, these rules can be handled with one SQL at one time [14].

```
Assume A = {A1, A2, …, Ap}, then the following SQL statement works:
CREATE VIEW TA
SELECT A1, A2, …, Ap, d, sum(support)
FROM (SELECT A1, A2, … Ap, d, count(*) support
FROM T
GROUP BY A1, A2, …, Ap, d)
GROUP BY A1, A2, …, Ap
HAVING count(*) = 1 and sum(support) >= s
```

Figure 4.2: SQL statement handling inconsistent and insufficient supported rules

The inner SELECT statement groups (merges) all duplicate tuples and counts their supports, while the outer SELECT statement removes all inconsistent tuples which have the same conditions but different decisions.

In Lines 7 and 8, tuples that are covered by current RB is discarded. A tuple is covered by RB if it is a super-tuple of a tuple in RB.

4.1.2 Handle Covered Rules

Another issue in applying this algorithm is how to handle rules covered by the existing rules. Known from the set theory, if the rule t=>s is an existing rule, then the extended t ∧p=>s is coverd by t=>s. For instance, there are rules as follows:

If C1=3 then Decision=3 (R1)

If C1=3 ∧ C2=3 then Decision=3 (R2)

Obviously, C1 value of 3 is adequate to induce the decision value of 3. Although C1=3 $\wedge$ C2=3 can also bring to a decision 3, condition C2=3 is redundant for the decision value 3. Rule R2 is also considered as covered by Rule R1.

To maintain an optimized rule set, all the rules covered by the existing rules should be excluded from the final result list. The SQL for eliminating covered rules is shown as below:

TA: the view contains the rules need to be checked.

R: the table contains existing rules.

```
CREATE VIEW RBTemp
Select * from
(Select   TA.A1, TA.A2, ..., TA.Ap,   TA.d, R.d as Flag as flag from TA left outer join R
on TA.A1 like R.A1
and TA.A2 like R.A2
...
And TA.Ap like R.Ap
And TA.d = R.d) as Temp
Where Temp.flag = NULL
```

Figure 4.3: SQL for Covered Rules

The inner SELECT statement joins the newly created rules with the existing rules. In Line 8, if a rule is not covered by existing rules, Temp.flag will be Null. Thus, all rules with empty right part are not covered and need to be inserted into the table of the existing rules.

4.2 Example of Implementation

To express the whole process of the algorithm explicitly, a simple MDT table was used as an example. It has 3 condition attributes {C1, C2, C3}. The value domain of decision attribute is 3; each is assigned correspondent support parameter W1, W2, W3 respectively.

During the first iteration, the candidate set of condition attributes are built with size 1 for each member element. For this example, the set is {(C1), (C2), (C3)}. After the candidate rule set is taken with condition attributes {C1}, the inconsistent and insufficient supported rule could be figured out immediately.

Table 4.1: Origin MDT table and decision rule for condition attribute C1

No	C1	C2	C3	wt	D1	D2	D3	W1	W2	W3
1	1	1	1	1	1	0	0	1	0	0
2	1	2	1	1	1	0	0	1	0	0
3	1	2	2	7	1	1	0	3	4	0
4	1	2	3	1	0	1	0	0	1	0
5	1	3	3	2	0	1	0	0	2	0
6	2	3	3	3	0	1	1	0	1	2
7	3	3	3	1	0	0	1	0	0	1
8	3	4	4	1	0	0	1	0	0	1

(A) MDT table

No.	C1	wt	D1	D2	D3	W1	W2	W3
1	1	12	1	1	0	5	7	0
2	2	3		1	1	0	1	2
3	3	2	0	0	1	0	0	2

(B) Rules decided by C1

It is clear that two tuples with the same condition attributes value inducing to the different decision attribute value, these two tuples are inconsistent. In the first rule, when C1 with value 1 the rule can induce to two decisions- D1 and D2, which has support number 5 and 7 respectively. Obviously, tuple 1 is inconsistent. The same reason, tuple 2 is inconsistent too. Thus, only Tuple 3 will remains in the target rule table.

Currently the rule set R is empty, therefore rules in Table 4.1(B) is not covered by any rule. Thus, the first rule in the target rule table is shown in Table 4.2.

Table 4.2: Decision Rule for {C1,d}

No.	C1	C2	C3	wt	D1	D2	D3	W1	W2	W3
1	3	-	-	2	0	0	1	0	0	2

After completing step 4 in the algorithm, only one rule passed the test procedure as listed in Table 4.2. The value of '-' in C2 and C3 means they are dispensable for rule pair {C1,D, support} = {3,3,2}. With one attribute value C1=3, it can be achieved that the unique decision equals to 3 with the support value of 2.

After checking all 1-item condition attribute candidate rules, all valid rules currently are displayed in Table 4.3.

Table 4.3: 1-item condition attribute rules

No.	C1	C2	C3	wt	D1	D2	D3	W1	W2	W3
1	3	-	-	2	0	0	1	0	0	2
2	-	1	-	1	1	0	0	1	0	0
3	-	4	-	1	0	0	1	0	0	1
4	-	-	1	2	1	0	0	2	0	0
5	-	-	4	1	0	0	1	0	0	1

Then, the next step is to find rules for 2-item condition attributes candidate set. The 2-item condition attributes set is {(C1, C2), (C1, C3), (C2, C3)}.

Table 4.4: 2-item condition attributes candidate rules

No.	C1	C2	C3	wt	D1	D2	D3	W1	W2	W3
1	1	1	-	1	1	0	0	1	0	0
2	1	3	-	2	0	1	0	0	2	0
3	3	3	-	1	0	0	1	0	0	1
4	3	4	-	1	0	0	1	0	0	1
5	1	-	1	2	1	0	0	2	0	0
6	1	-	3	3	0	1	0	0	3	0
7	3	-	3	1	0	0	1	0	0	1
8	3	-	4	1	0	0	1	0	0	1

No.	C1	C2	C3	wt	D1	D2	D3	W1	W2	W3
9	-	1	1	1	1	0	0	1	0	0
10	-	2	1	1	1	0	0	1	0	0
11	-	2	3	1	0	1	0	0	1	0
12	-	4	4	1	0	0	1	0	0	1

These 3 shadowed rules are formal decision rules necessary for the resulting rule set. Others are covered by the existing rules in Table 4.4.

If a tuple is the super-tuple of a tuple in existing rules, it is covered by these existing rules. For example, tuple 1 in Figure 4.8 is the attribute-decision pair ((C1, C2), Decision)= ((1,1),1), it covered by tuple 2 in Figure 4.7 (C2,Decision)= (1,1). Eventually, this step is to check 3-item condition attributes candidate set. It is proved that there are no 3-item condition attributes rules which are consistent and not covered by existing rules. Thus the final result of value reduction for Table 4.1(A) is shown as Table 4.5.

Table 4.5: Rules after value reduction

No.	C1	C2	C3	wt	D1	D2	D3	W1	W2	W3
1	3	-	-	2	0	0	1	0	0	2
2	-	1	-	1	1	0	0	1	0	0
3	-	4	-	1	0	0	1	0	0	1
4	-	-	1	2	1	0	0	2	0	0
5	-	-	4	1	0	0	1	0	0	1
6	1	3	-	2	0	1	0	0	2	0
7	1	-	3	3	0	1	0	0	3	0
8	-	2	3	1	0	1	0	0	1	0

If we use a threshold of support >1, thus only Tuple1, 4, 6 and 7 remain valid in Table 4.5.

CHAPTER V

ROUGH SET ASSOCIATION RULE MINING VALUE REDUCT ALGORITHM (RSAVR)

The main focus of this chapter is to present a new algorithm based on the association rule-mining and rough set theory like previous Ma's work [16]. Some paperThis algorithm was built on the foundation of association rule mining and rough sets theory which are discussed in the previous chapters.

The purpose of the proposed algorithm is to execute much less scan than HFVR only if pre-generated by the association rule.

5.1 Description of RSAVR Algorithm

The proposed algorithm can be summarized in the following steps:

```
Step 1:  create the MDT for decision table, let G=MDT
Step 2:  check if G is empty, if G is not empty continue Step 3 to step 6
Step 3:  Set k as 1, now $C_k = C_1$= all singleton attributes of condition attributes set
Step 4:  for each element C in $C_{k,}$ use function GenRuleTable to generate rule set $R_k$
Step 5:  run function ValidateRules to remove invalid rules from $R_{k,}$ remove C from
           $C_{k,}$ set R = R U $R_k$
Step 6:  Repeat Steps 4 to 5 until $C_k$ is empty
Step 7:  Increase K to K+1, generate Candidate rule Set $C_{k+1}$, G = G-[R]
Step 8:  Repeat Step 2 to 7 until G is empty
Step 9:  End
```

Figure 5.1: Brief steps of RSAVR

Compared to the algorithm described in Chapter 4, this proposed algorithm has similar steps in Step3 and Step 4 which can generate rules from the original data set. However, there are more differences between these two algorithms. In step 7, the candidate set is generated with apriori algorithm to increase the accuracy, reduce the time of database scanning. In this step, some candidate rules may emerge with low support. Nevertheless, the cut-down of candidate set may also cause some interesting and helpful rules excluded from the result rule set. Hence, step 8 is used to make sure all the valid rules can be collected without the exception.

The overview of the algorithm is shown as following [15].

```
begin
    TargetRuleTable = empty;
    G = MDT; //Target table
    while (G is not empty) do
    begin
        K = 1; //number of condition attributes
        $C_k$ = all singleton attributes of the set of condition  attributes;
        while ($C_k$ is not empty) do
        begin
            R = empty; //initialize temp rule set
            for each C in the candidate set $C_k$ do
            begin
            $R_k$ = GenRuleTable(G, C, s);  //Use the above query to generate rule set $R_k$;
            ValidateRules(MDT, $R_k$); //remove invalid rules
            if ($R_k$ is not empty)
            begin
                Remove C from $C_k$;
                R = R ∪ $R_k$;
             end;//if
        end;//for
        K++;
        $C_k$ = GenerateCandidates($C_k$, K); //self-join of two
        G = G – [R]; //Update G by removing instances covered by R
    end;// while $C_k$ candidate set is not empty
    Add R to TargetRuleTable;
    end; // while G is not empty
    return TargetRuleTable;
end;// VR_MDT
```

Figure 5.2: A high-level description of RSAVR

5.1.1 Rule Generation and Validation

Another feature in RSAVR algorithm is that rule generation and validation can be accomplished using the same group-by condition applying to SQL queries with different parameters. The basic template is given in the following.

```
select Condition_Attributes_List,
        d1=max(d1),..., dn=max(dn),
        w1=sum(w1),...,wn=sum(wn)
from Target_MDT
group by Condition_Attributes_List
having
(sum(w1)+...+sum(w3) > @minsupport) and
count(distinct cast(d1 as varchar(1))+
'#'+...+'#'+cast(dn as varchar(1)))=1
```

Figure 5.3: rule generation and validation

The parameter @minsupport is the support threshold that defines frequent item-set.

5.2 Example of Implementation for RSAVR Algorithm

In the first iteration, the 1-item candidate set is built from the original condition attributes. 1-item means only one condition attribute is input with specific value, other condition attributes can be any value (character '-' substituted for specific value in the example table). For example, the 1-item set here is {(C1), (C2), (C3)} in which each set member consists one attribute.

Table 5.1: Decision Rule induced by Condition Attribute C1

No.	C1	C2	C3	wt	D1	D2	D3	W1	W2	W3
1	3	-	-	2	0	0	1	0	0	2

After step 4 in is completed, same rule is obtained compare to HFVR. Other two candidate rules were eliminated because of the insufficient support in table.

Nevertheless, 'C1' will be removed according to the RSAVR algorithm from 1-item set because it is a frequent item which has support value 2 greater than minimum support 1.

Table 5.2: decision rule after detection for 1-item set

Tuple No.	C1	C2	C3	wt	D1	D2	D3	W1	W2	W3
1	3	-	-	2	0	0	1	0	0	2
2	-	1	-	1	1	0	0	1	0	0
3	-	4	-	1	0	0	1	0	0	1
4	-	-	1	1	1	0	0	2	0	0
5	-	-	4	1	0	0	1	0	0	1

After the detection for all members in 1-item set, C1, C2 and C3 are all excluded from 1-item set because they are frequent item with support greater than 1. Now there will be no attributes left in 1-item set. Thus 2-item candidate set generated based on 1-item set will be empty too.

Table 5.3: decision rule after 1-item set detection with minsup =2

Tuple No.	C1	C2	C3	wt	D1	D2	D3	W1	W2	W3
1	3	-	-	2	0	0	1	0	0	2

Table 5.3 shows rules generated from 1-item set with minimum support value 2 at the beginning of the algorithm. Only C1 will be removed from 1-item set. 2-item set will be generated from current 1-item set {(C1), (C2)}. According to Apriori algorithm, candidate 2-item set is {C1, C2}. Compare to HFVR, another 2 detection item {(C1, C3), (C2, C3)} are reduced. Hence, the algorithm can be more efficient with less candidate rule detection.

CHAPTER VI

EXPERIMENTAL RESULT

In this chapter, the different experiments designed to measure the performance of RSAVR algorithm compared to HFVR algorithm are discussed.

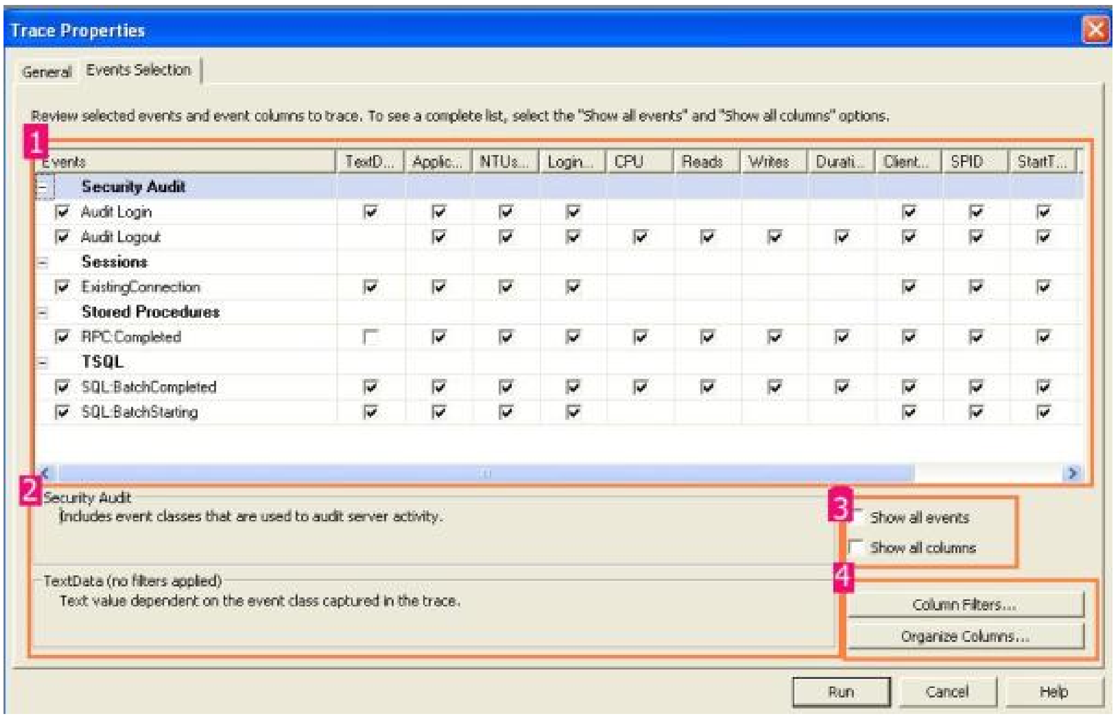
Section 6.1 describes the environment in which the performance experiments were executed, including both the hardware and the software. It also talks about the data set used to run the test experiments.

Then, in following section 6.2, it describes the classification of the experiments that measure the performance of RSAVR relative to HFVR algorithm for different sizes of reduction tests. Section 6.3 lists the result of the experiments and analyzes these results.

6.1 The Experimental Environment

In the experiments, the testing programs were deployed on the server cluster with Quad core AMD Opteron™ CPU, and 32 GB memory. The rule reduct procedure was implemented as MS SQL stored procedure, running on MS SQL SERVER 2005.

The original format of data is text file. All the implemented programs are targeted in the relational table. BCP was used (Bulk Copy Program) to convert the data from text file to data in the table or reversely. In order to monitor the running time of the algorithm programs, SQL Profiler was used to trace events inside SQL Server. SQL Profiler is a graphical tool provided by SQL Server that allows user to monitor events recorded in SQL Server instance. With the help of SQL Profiler, the exact starting time and end time of the stored procedures, and the evidence of slow-executing queries were collected. Thus, a complicated t-sql program was analyzed with the help of SQL Profiler (see figure 6.1).


Trace Properties

General | Events Selection

Review selected events and event columns to trace. To see a complete list, select the "Show all events" and "Show all columns" options.

1

Events	TextD...	Applic...	NTUs...	Login...	CPU	Reads	Writes	Durati...	Client...	SPID	StartT...
Security Audit											
✓ Audit Login	✓	✓	✓	✓					✓	✓	✓
✓ Audit Logout		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Sessions											
✓ ExistingConnection	✓	✓	✓	✓					✓	✓	✓
Stored Procedures											
✓ RPC:Completed	☐	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
TSQL											
✓ SQL:BatchCompleted	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
✓ SQL:BatchStarting	✓	✓	✓	✓					✓	✓	✓

2 Security Audit
Includes event classes that are used to audit server activity.

TextData (no filters applied)
Text value dependent on the event class captured in the trace.

3 ☐ Show all events
☐ Show all columns

4

Figure 6.1 filter the events going to be monitored

For the resource dataset, IDS (Intrusion Detection Systems) dataset was selected to evaluate two algorithms. It is made available by MIT Lincoln labs and can be downloaded from UCI Machine learning Repository. The repository displays as http://archive.ics.uci.edu/ml/datasets.html [18].

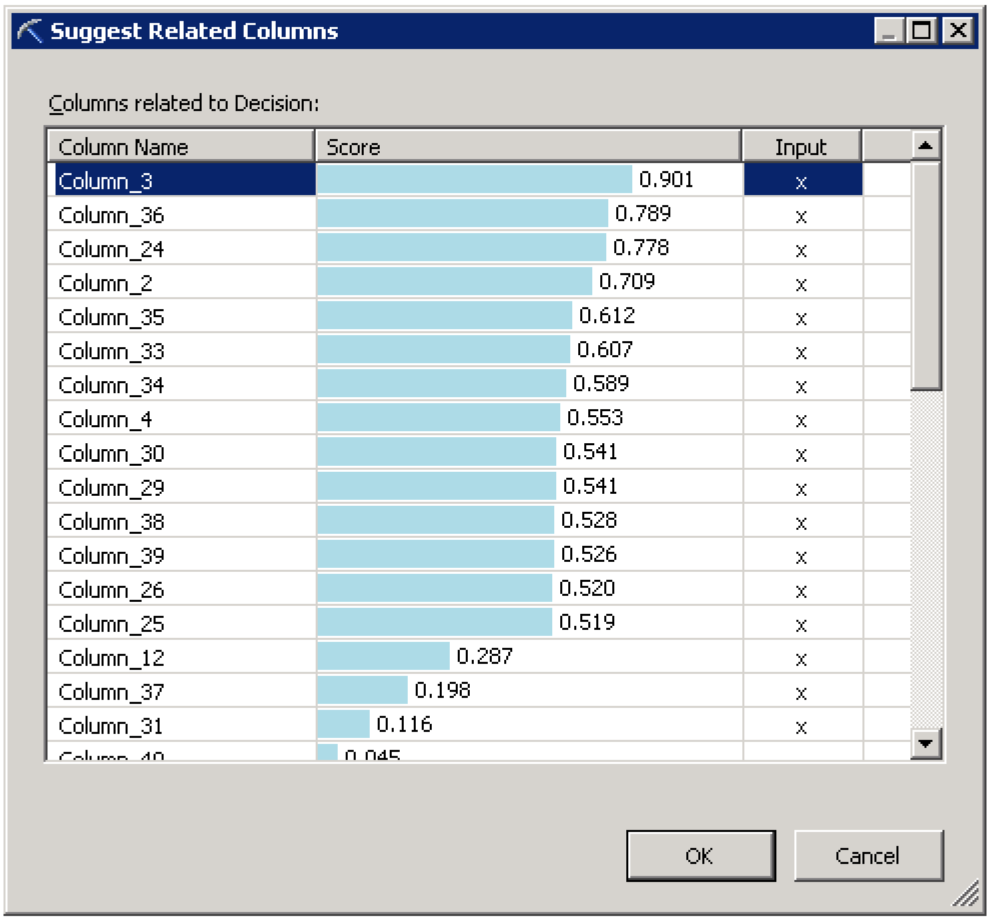
Since 1999, this IDS dataset (KDD' 99) has been widely used in many of the data mining researches. The dataset contains about 5 million connection records and the total file size is 772MB.

Totally, this dataset consists of 42 attributes. 41 are condition attributes and the 42th is decision attribute. In these 41 condition attributes, 34 are numerical attributes and 7 are categorical attributes.

6.2 Pre-processing

The beginning of the experiment was preprocessing data included in the test. As introduced in Chapter 3, raw dataset contains incomplete, incorrect data that should be revised before experimental processing.

In this research, SQL Server Analysis Services provides powerful tools to help us construct decision tree. For the same purpose, these tools can be useful to generate a healthy, ready to use data set for our testing algorithm. Figure 6.2 shows the attributes most likely related to decision that is identified by SQL Server Analysis Service.



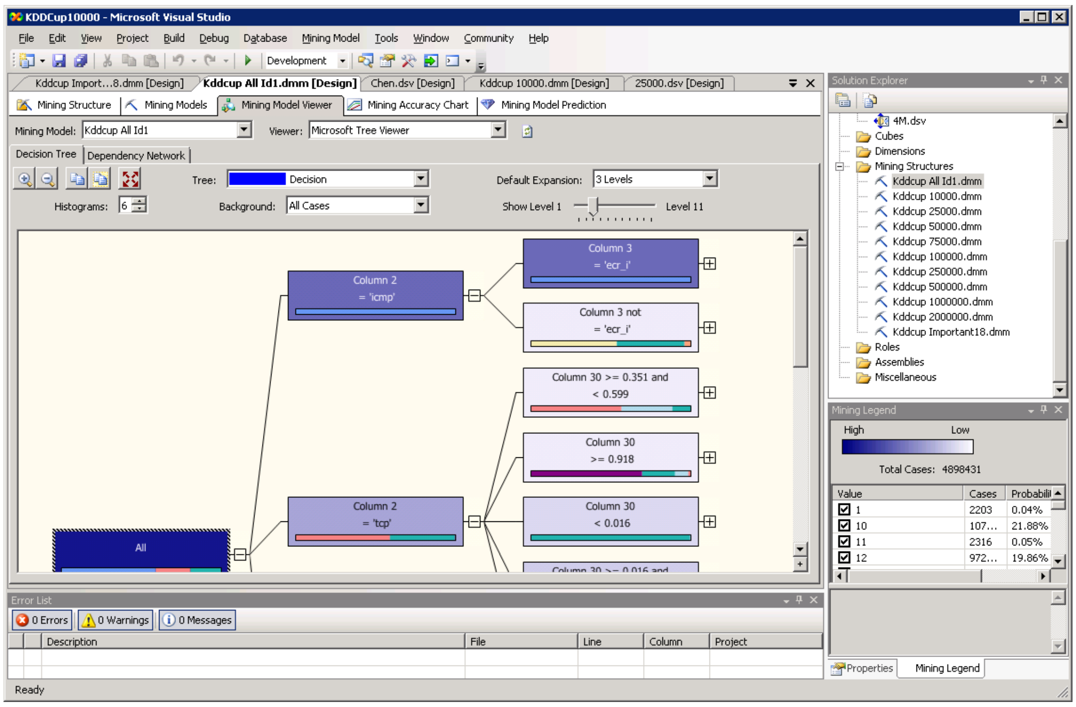
Columns related to Decision:

Column Name	Score	Input
Column_3	0.901	x
Column_36	0.789	x
Column_24	0.778	x
Column_2	0.709	x
Column_35	0.612	x
Column_33	0.607	x
Column_34	0.589	x
Column_4	0.553	x
Column_30	0.541	x
Column_29	0.541	x
Column_38	0.528	x
Column_39	0.526	x
Column_26	0.520	x
Column_25	0.519	x
Column_12	0.287	x
Column_37	0.198	x
Column_31	0.116	x
Column_40	0.045	

Figure 6.2: Attributes related to decision

With the help of SQL Server Analysis Service, 18 condition attributes and 1 decision attribute, columns 2, 3, 4, 12, 24, 25, 26, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39 and 42 from original data set, were considered as critical ones to go through next step.

From decision tree algorithm, the guide for discretization was prepared quickly. In this step, all continuous attributes were split into intervals. Then, SQL statements were applied to make discretization according to these intervals. A snapshot of decision tree generated by SQL Server is shown in Figure 6.3.



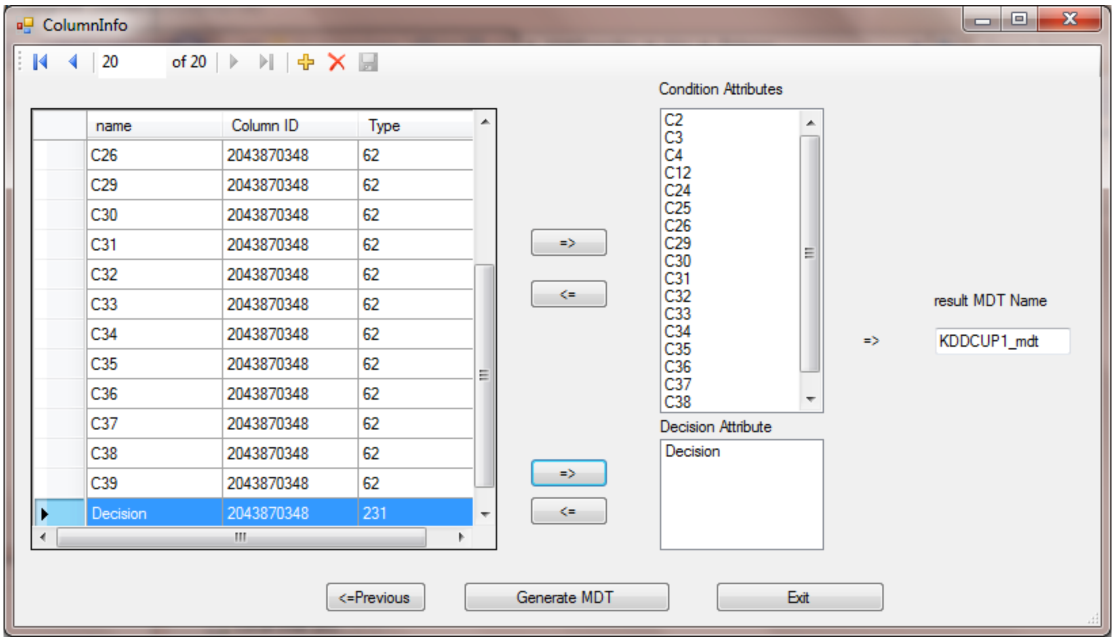
Mining Model: Kddcup All Id1
Viewer: Microsoft Tree Viewer
Tree: Decision
Default Expansion: 3 Levels
Histograms: 6
Background: All Cases

Total Cases: 4898431

Value	Cases	Probabili
☑ 1	2203	0.04%
☑ 10	107...	21.88%
☑ 11	2316	0.05%
☑ 12	972...	19.86%

Figure 6.3: View of SQL Server Decision Tree

In next step, MDT for IDS data set was generated by the algorithm proposed in [18]. A snapshot of the GUI for generating MDT is shown in Figure 6.4. The comparison of storage space can be processed after this step.



name	Column ID	Type
C26	2043870348	62
C29	2043870348	62
C30	2043870348	62
C31	2043870348	62
C32	2043870348	62
C33	2043870348	62
C34	2043870348	62
C35	2043870348	62
C36	2043870348	62
C37	2043870348	62
C38	2043870348	62
C39	2043870348	62
Decision	2043870348	231

Condition Attributes
C2
C3
C4
C12
C24
C25
C26
C29
C30
C31
C32
C33
C34
C35
C36
C37
C38

Decision Attribute
Decision

result MDT Name
KDDCUP1_mdt

Figure 6.4: Generating MDT

6.3 Classification of the Experiments

The experiments in this study compared the performance of RSAVR and HFVR on different scales of the database. The size of the testing database was varied on ten different scales from 10, 000 to more than 4,000,000 records. Each n-th scale set contains 2 times of records of the previous scale. The scale settings used for the experimental database are listed in Table 6.1.

Table 6.1: Scale of database in the experiment

Test	1	2	3	4	5	6	7	8	9	10
Scale (Records in Database)	9,427	18,854	37,708	75,416	150,833	301,666	603,332	1,206,664	2,413,328	4,826,656

Another critical and concerning issue is how the threshold of support affects the performance. For this purpose, the support threshold was increased in small steps until the performance changed remarkably. Therefore, the effect of the support threshold on the performance was compared for each specific algorithm.

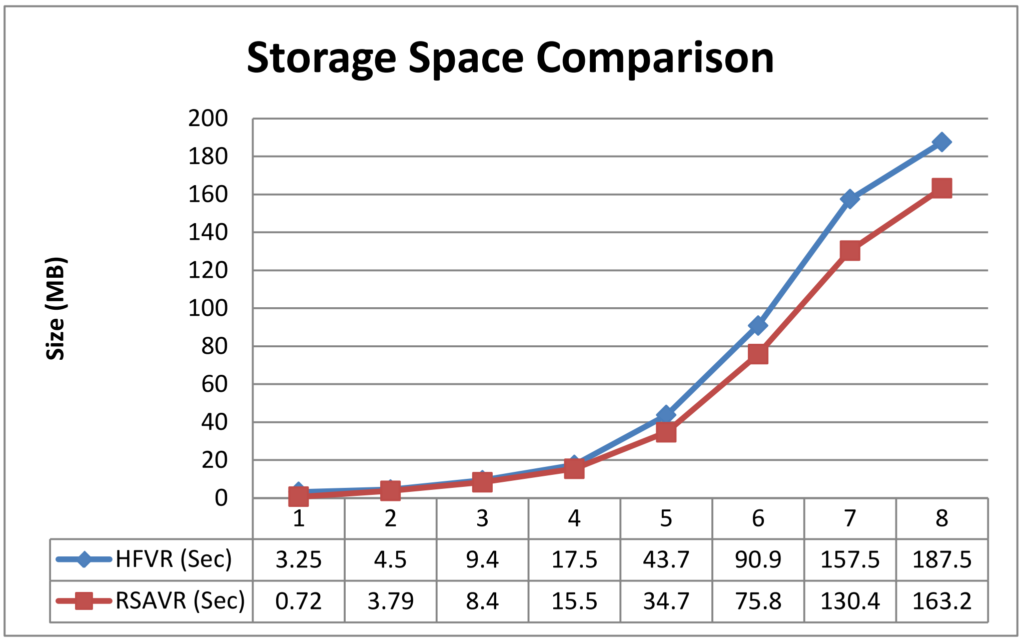
6.4 Test and Result

In Table 6.2 the classification of performance obtained by applying different approaches to the IDS dataset was gathered. For each approach, the dataset was analyzed with the same approach for multiple times to ensure there was no major error. Fortunately, the approximate data obtained showed the great confidence that both the server and the program were running stably.

Table: 6.2 Experimental result of 19 attributes

Size of Decision Table (MB)	Size of MDT (MB)	Records in DT (Rows)	Time of HFVR in MDT (sec)	Time of HFVR MDT supp > 2 (sec)	Time of RSAVR in MDT (sec)	Time of RSAVR in MDT Supp >2 (Sec)	Time of Decision Trees (sec)
1.547	0.016	9,427	1.78	3.25	3.74	0.72	3
3.078	0.031	18,854	5.97	4.5	6.85	3.79	6.2
6.148	0.039	37,708	12.9	9.4	16.5	8.4	18.4
12.266	0.055	75,416	24.2	17.5	25.75	15.5	26
24.289	0.056	150,833	58.2	43.7	49.6	34.7	62
48.383	1.305	301,666	149.6	90.9	125.7	75.8	91
96.484	1.719	603,332	232.4	157.5	194.8	130.4	204
192.68	2.785	1,206,664	356.8	187.5	224.4	163.2	250
385.64	3.184	2,413,328	469.4	260.8	252.6	197.6	335
783.95	7.016	4,826,656	720.2	530.4	496.9	325.9	771

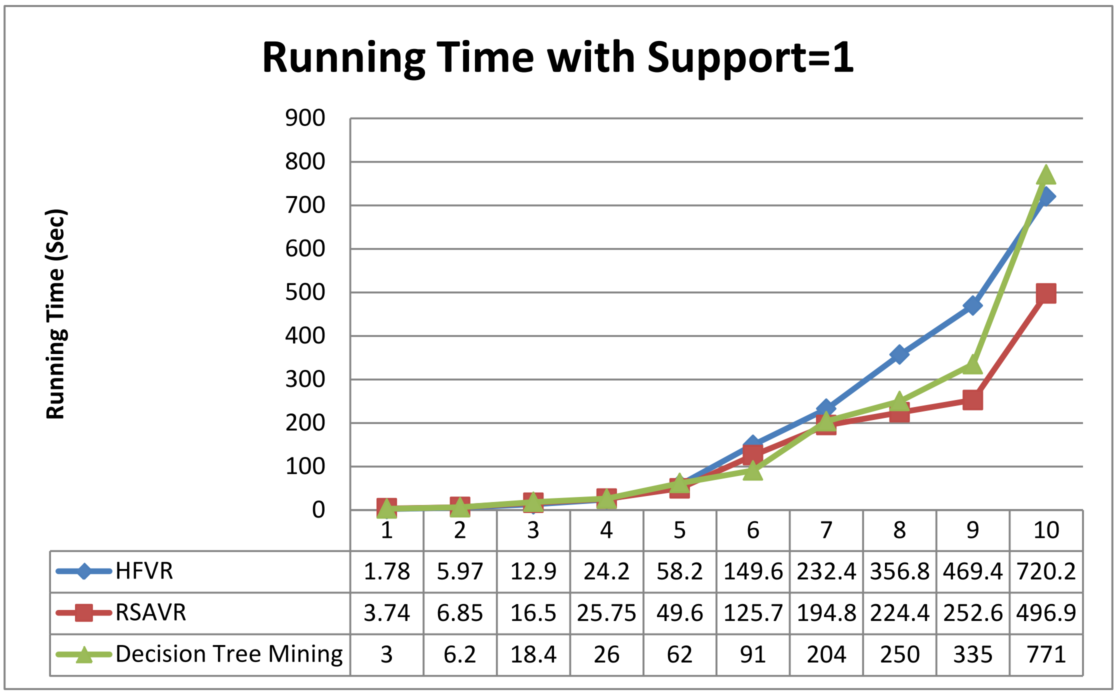
6.4.1 Measuring Performance for Different Scale of Database



	1	2	3	4	5	6	7	8
HFVR (Sec)	3.25	4.5	9.4	17.5	43.7	90.9	157.5	187.5
RSAVR (Sec)	0.72	3.79	8.4	15.5	34.7	75.8	130.4	163.2

Figure 6.5: Space reduction of MDT compare to decision table

In Figure 6.5, the size of the storage space (indication of the data reduction rate) was investigated for both the decision table and MDT while varying the dataset. As shown in the figure, for each data set scale MDT consumed much less storage space than what decision table did. To store the whole data set, only 24M was used by MDT, only one tenth of decision table. As expected, the reduction rate increased as the data scale increased.



	1	2	3	4	5	6	7	8	9	10
◆ HFVR	1.78	5.97	12.9	24.2	58.2	149.6	232.4	356.8	469.4	720.2
■ RSAVR	3.74	6.85	16.5	25.75	49.6	125.7	194.8	224.4	252.6	496.9
▲ Decision Tree Mining	3	6.2	18.4	26	62	91	204	250	335	771

Figure 6.6 running time of HFVR, RSAVR and Decision Tree

Figure 6.6 displayed the difference of running time used by HFVR, RSAVR and decision tree while data set varies. In figure 6.3, it is observed that

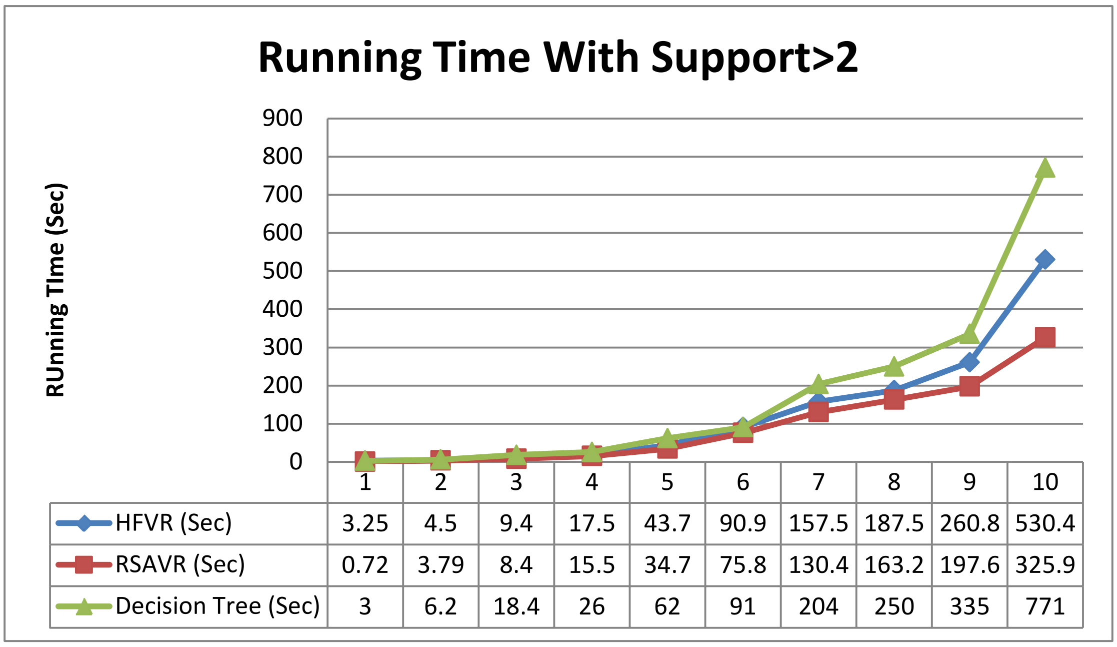
i. From data scale 1 to 4, HSVR and RSAVR had similar running time. It took RSAVR slightly more time to finish the analysis, barely one quarter more than HFVR.

ii. In scales 5 and 10, the running time of HFVR exceeded what RSAVR used significantly. As the amount of data increased, the gap of running time between HFVR and RSAVR becomes even larger.

iii. Compare to Decision tree mining, RSAVR spent less time in all scales except No.6. But HFVR need more running time in most scales.

iv. HFVR had better performance in small scale database such as data scale 1 to 4; however, as the amount of data grew, RSAVR showed much better performance.

6.4.2 Measuring performance for different support threshold



	1	2	3	4	5	6	7	8	9	10
HFVR (Sec)	3.25	4.5	9.4	17.5	43.7	90.9	157.5	187.5	260.8	530.4
RSAVR (Sec)	0.72	3.79	8.4	15.5	34.7	75.8	130.4	163.2	197.6	325.9
Decision Tree (Sec)	3	6.2	18.4	26	62	91	204	250	335	771

Figure 6.7 running time of HFVR/RSAVR(support>2)

The running time of HFVR with various support thresholds was displayed in figure 6.7. It indicated that the support parameter has great impact to the experiment result.

Overall, both HFVR and RSAVR achieved better performance than decision tree mining. Nevertheless, RSAVR could get much better performance than HFVR in all data scales with minimum support greater than 2 being applied, which removed small groups of instances from MDT.

Especially, for experiments on the larger scales (9-10), running time for HFVR in MDT and decision tree grew significantly faster than that in RSAVR with support threshold more than 2.

CHAPTER VII

CONCLUSION AND FUTURE WORK

7.1 Conclusion

This research is about reducing irrelevant data from rules based on rough set theory tools implemented using relational databases.

With KDD methodology, two algorithms were implemented for value reduction. Furthermore, both algorithms applied the minimum support threshold to remove small group of instances in the MDT.

Base on the experiments, parameters may bring large impact to algorithm's performance. First, with the same algorithm, performance varies in different scale of dataset. In the experiments with small data scale, HFVR provides better performance than RSAVR. However, in the test with large dataset scale, it showed the reverse result. By applying Association rules in generating candidate rule set, RSAVR remarkably reduced the number of scans over data set. Because this scan cost much more time on large dataset, RSAVR's privilege became more obvious as the data scale increased. Thus, it was concluded that running time of RSAVR was significant less than that for HFVR in large data set.

Secondly, the minimum support parameter also brings outstanding changes to the performance. With the incensement of minimum support parameter, the performance of both algorithms were improved.

7.2 Future work

Several questions remain unanswered and are recommended for the future work:

- Minimum support threshold is the major parameter of the algorithms. In this study, the minimum support threshold had to be increased step by step to reach the point of better performance and accuracy; and a simpler solution needs to be developed to detect this balanced minimum support. In this case, it might be more appropriate to provide an alternative method to combine with value reduction algorithms.
- Accuracy is another important aspect of value reduction algorithm. However, it is not discussed much in the thesis. It is also suggested the proper method should be established to examine the accuracy of the algorithms.

REFERENCES

[1] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth: From Data Mining to Knowledge Discovery: An Overview. Advances in Knowledge Discovery and Data Mining, MIT Press (1996)

[2] Pawlak, Z.: Rough sets: basic notion. Int. J. of Computer and Information Science 11, 344–356 (1982)

[3] Pawlak, Z.: Rough sets and decision tables. In: Skowron, A. (ed.) SCT 1984. LNCS, vol. 208, pp. 186–196. Springer, Heidelberg (1985)

[4] Zdzislaw Pawlak, Rough Sets: Theoretical Aspects of Reasoning about Data, Kluwer Academic Publishers, Norwell, MA (1992)

[5] Pawlak, Z., Grzymala-Busse, J., Slowinski, R., Ziarko, W: Rough sets, Communication of ACM,. 38(11), 89-95 (1995).

[6] Xiaohua Hu.: Knowledge Discovery in Databases: an Attribute-Oriented Rough Set Approach. PhD thesis, University of Regina, (1995)

[7] Grzymala-Busse, J.W.: Learning from examples based on rough multisets. Proc. of the 2nd Int. Symposium on Methodologies for Intelligent Systems, Charlotte, North Carolina, October 14-17, pp. 325-332 (1987)

[8] Chan, C.-C.: Learning rules from very large databases using rough multisets. LNCS Transactions on Rough Sets, (J. Peters, A. Skowron, J.W. Grzymala-Busse, B. Kostek, R.W. Swiniarski, M. Szczuka Eds.), Vol. 1, pp. 59 – 77, Springer – Verlag Berlin Heidelberg (2004)

[9] Pawlak, Zdzislaw.: Some Issues on Rough Sets. In: Peters, J.F., et al. (Eds.) Transactions on Rough Sets I. LNCS, 3100, pp. 1—58 Springer, Heidelberg (2004)

[10] Agrawal, R.; Imieliński, T.; Swami, A.: Mining association rules between sets of items in large databases. Proceedings of the 1993 ACM SIGMOD international conference on Management of data - SIGMOD '93. p. 207 (1993)

[11] Rakesh Agrawal; Ramakrishnan Srikant: Fast algorithms for mining association rules. In Jorge B. Bocca, Matthias Jarke, and Carlo Zaniolo, editors, Proc. 20th Int. Conf. Very Large Data Bases, VLDB, pages 487–499. Morgan Kaufmann, 12–15 (1994)

[12] C. Borgelt.: Efficient implementations of apriori and eclat. In Proceedings of the FIMI'03 Workshop on Frequent Itemset Mining Implementations, Melbourne, Florida, USA, November, CEUR Workshop Proceedings (2003)

[13] Jian Pei; Jiawei Han; Runying Mao: CLOSET: An efficient algorithm for mining frequent closed itemsets. In ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery, pages 21–30 (2000)

[14]Lin, T.-Y. and Jianchao Han, "High Frequent Value Reduct in Very Large Databases," LNCS Vol. 4482, Springer –Verlag Berlin Heidelberg, 346 – 354, (2007)

[15] Chien-Chung Chan; Chen Chen; Garikapati, V.: Finding Value Reducts Using Rough Multisets. Granular Computing (GrC), 2010 IEEE International Conference on , vol., no., pp.85,89, 14-16 Aug. 2010 doi: 10.1109/GrC.2010.150 (2010)

[16]Ma, Yu-Liang and Wen-jun Yan: Value reduction algorithm in rough sets based on association rules support. Journal of Zhejiang University, Vol. 7, Supplement 2, August, 2006, 219 – 222. (2006)

[17]Seelam, Uday; C.-C. Chan: A study of data reduction using multiset decision tables. Proceedings of IEEE Granular Computing 2007, November 2 – 4, 2007, San Jose, USA, pp. 362-367UCI data set (2007)

[18] Hettich, S. and Bay, S. D.: The UCI KDD Archive [http://kdd.ics.uci.edu]. Irvine, CA: University of California, Department of Information and Computer Science. (1999)