12-10-2015

Efficient Methods for Mining Association Rules from Uncertain Data

Manal Hamed Alharbi
manal.alharbi@uconn.edu

Efficient Methods for Mining Association Rules from Uncertain Data

Manal Hamed Alharbi, PhD

University of Connecticut 2015

Association rules mining is a common data mining problem that explores the relationships among items based on their occurrences in transactions. Traditional approaches to mine frequent patterns may not be applicable for several real life applications. There are many domains such as social networks, sensor networks, protein-protein interaction analysis, and inaccurate surveys where the data are uncertain. As opposed to deterministic or certain data where the occurrences of items in transactions are definite, in an uncertain database, the occurrence of an item in a transaction is characterized as a discrete random variable and thus represented by a probability distribution. In this case the frequency of an item (or an itemset) is calculated as the expected number of occurrences of the item (itemset) in the transactions. In this research work, we present efficient computational algorithms for three important problems in data mining involving uncertain data. Specifically, we offer algorithms for weighted frequent pattern mining, disjunctive association rules mining, and causal rules mining, all from uncertain data. Even though

algorithms can be found in the literature for these three versions of rules mining, we are the first ones to address these problems in the context of uncertain data.

Efficient Methods for Mining Association Rules from Uncertain Data

Manal Hamed Alharbi

M.S., King Abdul-Aziz University, Saudi Arabia, 2008

B.A., King Abdul-Aziz University, Saudi Arabia, 1998

A Dissertation

Submitted in Partial Fulfillment of the

Requirements for the Degree of

Doctor of Philosophy

At the

University of Connecticut

2015

APPROVAL PAGE

Doctor of Philosophy Dissertation

Efficient Methods for Mining Association Rules from Uncertain Data

Presented by

Manal Hamed Alharbi, B.S., M.S.

Major Advisor _____

Sanguthevar Rajasekaran

Associate Advisor_____

Reda A. Ammar

Associate Advisor_____

Chun-Hsi Huang

University of Connecticut

2015

PREFACE

The Doctoral Dissertation “Efficient Methods for Mining Association Rules from Uncertain Data” is part of the research conducted at the Department of Computer Science and Engineering, University of Connecticut. The computing facilities at the department of Computer Science and Engineering were used for the purpose of this research. No human or animal subjects were involved in this research.

ACKNOWLEDGEMENTS

First and foremost, I must to praise and thank Allah (God) Almighty, who always helped and blessed me with his innumerable pure mercy in all my life.

My great praise, deepest appreciation and acknowledgment go to my major advisor Dr. Sanguthevar Rajasekaran for his incredible guidance and extraordinary intellectual support all the time of research and writing of this thesis. During my studies and research under his guidance I saw literally how being a smart successful teacher does not contradict with being a humble person. His patience, motivation and mentoring were the cornerstone to achieve my goal.

I would like to extend my sincere grateful, fulfillment, and gratitude to Dr. Reda Ammar who opened many doors for me as well as many of my colleagues and provided us the opportunity to be a part of this great research community. He was a father figure for all his students, and he spared no effort to help us in every aspect in this extraordinary journey in our lives. I would like to thank Dr. Chun-Hsi Huang for kindly agreeing to be on the panel of defense Committee members.

Thanks to the unknown soldier behind each success in my life, to my role model, to my father who was all his utterances, and deeds literally a reflection of the values and ideals that we read on books' lines. Thanks to tribe of women, to my great teacher, to my mother who was a housewife and dedicated all her life to us. She had never eaten until

we all finished, takes care of all her 14 kids, prays to Allah to continue his blessing on her house, and all other houses in every place in this world. I am extremely thankful to my family for their constant material and moral support, and for their unconditional love. I am grateful to have such loving sisters (Fawziah, Sameera, Huda, Eman, Nawal, Nisreen, Noha, Wafa, and Fatmah). My research would not have been possible without their help, and best wishes. Thanks to my dearest brothers (Yaser, Osama, Talal, and Mohammed) - you are always there for me no matter what I do or say. Special thanks to my wonderful Grandmother Nassra, to my precious brothers in law (Abdul Jabbar, Adel) who passed away during the completion of this work.

Thanks to my friends for their encouragement. I would like to especially thank my best friend Huda Alhazmi for knowing my weakness and showing me my strengths. She was always there, shared all my moments with me, and stood by me through the good and bad times.

Last but not least, words are not sufficient to appreciate the person who has a great impact on my life, for making my life much happier, for granting me the positive energy that keep pushing me to finish this research.

Dedicated To

My father Hamed Eid Alharbi

Who taught me to smile in front of hardness, to value myself when lightning strikes me, to fight for my rights, to chase my dreams, and to appreciate blessings of life.

To my father when he always told me…

The one who walks on the right track would ultimately arrive.

Table of Contents

LIST OF FIGURES

LIST OF TABLES

Chapter 1

Introduction and Motivation

Frequent pattern mining is a well-studied problem that aims to discover the relationships among items based on their occurrences in transactions. Due to the growth of applications that involve uncertain data, traditional approaches to mine frequent patterns may not be applicable for several real life applications. In recent years, quality research has been conducted to associate the occurrences of items in uncertain databases for applications like social networks, sensor networks, protein-protein interaction analysis and inaccurate surveys. Uncertainty could also arise from masking of data for privacy concerns. Unlike certain data where the occurrences of items in transactions are definite, in an uncertain database, we are not sure about the items that occur in any transaction. Instead, we only know a probability for each possible item that this item belongs to a given transaction. In this thesis, we focus on three important problems in data mining involving uncertain data. Specifically, we propose two algorithms for weighted frequent patterns mining, a disjunctive association rules mining algorithm, and two algorithms for discovering both conjunctive and disjunctive causal rules mining, all from uncertain data. In spite of the fact that some algorithms have been introduced in

the literature for these three versions of rules mining, we believe that this is the first work that addresses these problems in the context of uncertain data.

Weighted frequent pattern mining is the problem that introduces the importance of an item as a significant factor besides its frequency. Unlike the typical frequent pattern mining methods that treat items as equally important, here we consider the case when each item has a different level of importance. The importance of weight-based pattern mining approach can be felt in many domains such as, biomedical data analysis where the causes of most diseases are not only one gene but a combination of genes; web traversal pattern mining where the impact of each web page is different; and so on. Thus, many algorithms have been proposed for weighting items based on their significance. To the best of our knowledge, no algorithms have been proposed in the literature for mining from uncertain weighted data. In this thesis we investigate the problem of mining from uncertain weighted data and present theoretical and experimental results for the proposed methods.

Disjunctive association rules mining is the problem when the antecedent and the consequent are sets of disjunctions (of items). A typical algorithm for association rules mining can be thought of as a conjunctive rule. There are many applications where rules with the antecedent as well as the consequent consisting of disjunctions of item sets might be relevant. The problem of generating disjunctive rules has been studied well. All the existing works on disjunctive rules thus far have been carried out to find disjunctive rules on databases without uncertainty. No work has been done on identifying disjunctive rules from uncertain data. The problem of disjunctive rules mining from uncertain data has numerous applications especially in biology, medicine, and bioinformatics. One of the

challenges in generating disjunctive rules lies in the need to explore a large collection of possible antecedents and consequents. Existing algorithms have large run times. In this research work, we propose an efficient algorithm for mining disjunctive rules from uncertain data.

Causal rules mining is the problem that mainly aims to discover profound relationships such as a change of antecedent is the cause for a change in the consequent. Traditional association rules mining algorithms identify the relationships among variables. By using support –confidence framework, we can show whether the antecedent and the consequent of a rule are related or not in general. However, associations do not necessarily signify causation. There are many cases where if the causes are predicted, we can easily avoid the consequences. The importance of causal relationships from uncertain data can be felt in many areas, such as economics, physical, behavioral, medical, biological, and social sciences. Statisticians have conducted many previous works in the direction of identifying causal rules. Also, partial association tests have been integrated along with association rule mining to minimize the exponential runtimes with the traditional statistical approaches of discovering Causal relationships. In this thesis we propose novel algorithms for finding both conjunctive and disjunctive Causal rules from uncertain transactional databases.

The rest of the dissertation is divided into five sections. In chapter 2, we present a summary on association rules mining algorithms, and discuss its most important algorithms for both certain and uncertain data. Also, we highlight the relative advantages and shortcomings of the different existing algorithms. In Chapter 3, we propose new algorithms for frequent itemsets mining on weighted uncertain data. In Chapter 4, we

present a novel algorithm for disjunctive rules mining from uncertain data. In Chapter 5, we propose new algorithms for both conjunctive and disjunctive Causal rules mining from uncertain data. Finally, we present our conclusions and describe future work in Chapter 6.

Chapter 2

Association Rules Mining From Uncertain Data

2.1 Traditional Association Rules Mining

Association rules mining is an important and well-researched problem in data mining which aims to find hidden relationships among items in large databases. Association rules mining algorithms have proven to be quite useful in many diverse fields including Web usage mining, intrusion detection, bioinformatics, etc. One of the first algorithms proposed for this problem was by Agrawal, et al. [1]. They introduced the idea for generating association rules from large scale transaction data in the context of the market basket problem. For example, a rule like $\{\text{milk}\} \Rightarrow \{\text{diaper}\}$ from a supermarket data indicates that when a person buys milk then (s)he is also likely to buy diaper. Knowing such information can be very useful for making proper decisions that increase the efficiency and reduce the cost associated with marketing activities such as product placements and promotional pricing [2].

Definition 2.1: A typical transactional dataset T consists of set of transactions where each transaction t comprises of a number of unordered items. The problem of

association rules mining can be stated as follows [1]: let the set of all possible items be $I = \{i_1, i_2, \cdots, i_m\}$. Each transaction t can be thought of as a subset of I. A subset of items is also known as an itemset. A $k - itemset$ is an itemset with k items. An association rule is nothing but an implication of the form $X \Rightarrow Y$, where $X \subset I$, $Y \subset I$, and $X \cap Y = \emptyset$. Here X and Y are defined as the antecedent and consequent of the rule, respectively.

Table 2.1shows an example. Table 2.1 in the left shows the itemset in binary format, where 1 signifies presence, and 0 signifies absence of any item in a transaction. Table 2.1 on the right shows a different representation of the itemsets in the transactions.

Table 2.1: Transactional Database

Transaction ID	milk	egg	diaper	beer
1	1	1	1	0
2	1	0	1	0
3	0	1	0	1
4	1	1	0	1
5	0	1	1	1

≡

Transaction ID	Items
T_1	milk, egg, diaper
T_2	milk, diaper
T_3	egg, beer
T_4	milk, egg, beer
T_5	milk, egg, beer

Support and confidence of a rule are two measures that signify the importance of the rule.

Definition 2.2: Support of a rule $X \Rightarrow Y$ is defined as the fraction of transactions that contain both X and Y. $\text{support}(X \cup Y) = \frac{number\ of\ records\ that\ contain\ X \cup Y}{total\ number\ of\ records\ in\ the\ database}$.

Example 2.1: the support of the rule $\{\text{milk}\} \Rightarrow \{\text{diaper}\} = \frac{2}{5} = 0.4 = 40\%$.

Definition 2.3: Confidence of a rule is defined as the number of transactions in which both X and Y occur divided by the number of transactions in which X occurs. $Confidence(X \Rightarrow Y) = \frac{\sup(X \cup Y)}{\sup(X)}$.

Example 2.2: the confidence of the rule $\{\text{milk}\} \Rightarrow \{\text{diaper}\} = \frac{0.4}{0.8} = 0.5 = 50\%$.

From among all such possible implications, only some of them could be of interest to us. Rules of importance are normally characterized with two parameters; $minsup$, and $minconf$, where the support of the rule implies that $\text{support}(X \cup Y)$ has to be at least $minsup$ and confidence of the rule stipulates that $confidence(X \Rightarrow Y)$ has to be at least $minconf$. Any itemset is said to be frequent if its support is at least $minsup$. Both $minsup$ and $minconf$ thresholds are user-defined values $\in [0,1]$

An important question is: How can we determine the best value for $minsup$ and *minconf* thresholds? In fact, there is no straightforward answer for this question and generally speaking, $minsup$ threshold is usually chosen through trial and error. Setting $minsup$ with a low value may lead to the generation of too many uninteresting frequent itemsets, and an increase in the time complexity. However, setting $minsup$ with a high value will improve the run time but may lead to the generation of no (or a very low number of) frequent itemsets.

Philippe in his PhD thesis [3, 4] suggested a mathematical function that allows one to modify the $minsup$ threshold dynamically based on the database size. He used this equation: $= (e^{(-aN-b))} + c$, where a, b, c are constants, and N represents the number of transactions.

In this thesis, we used the trial and error method. We set the $minsup$ parameter with a low value and then we gradually increase it until we extract a sufficient number of interesting patterns within a reasonable amount of time. For the $minconf$, we need to have a value of at least 50% for a rule to be interesting.

2.1.1 Traditional Frequent Patterns Mining Algorithms

A typical algorithm for association rules mining consists of two phases. In the first phase all the frequent itemsets are identified by searching through all possible Itemsets. Thus, it requires $2^n - 1$ time where n is the number of items in I. In the second phase, the frequent itemsets are used to generate association rules. The second phase is relatively simpler compared to the first phase and hence researchers mostly tackle the problem of identifying frequent itemsets. One of the most useful strategies that can be used to reduce the number of generated candidates is the $downward\ closure\ property$ [1]. $Downward\ closure\ property$ states that any subset of a frequent itemset is also frequent which implies that any superset of an infrequent itemset is also infrequent.

Example 2.3: Table 2.2, and Table 2.3 illustrate the strategy of finding association rules from large Itemsets, assume $minsup = 40\%$, and $minconf = 60\%$. For the transactional database in table 2.1, table 2.2 shows the frequent itemsets, and table 2.3 shows the association rules.

Table 2.2: Support for Itemsets for example 2.3

Itemset	**support (s)**	***minsup*=40%**
milk	80 %	> *minsup*
egg	80 %	> *minsup*
diaper	40 %	> *minsup*
beer	60 %	> *minsup*
milk, egg	40 %	> *minsup*
milk, diaper	40 %	> *minsup*
milk, beer	40 %	> *minsup*
egg, diaper	40 %	> *minsup*
egg, beer	50 %	> *minsup*
diaper, beer	0 %	< *minsup*
milk, egg, diaper	20 %	< *minsup*
milk, egg, beer	40 %	> *minsup*
milk, diaper, beer	0 %	< *minsup*
egg, diaper, beer	0 %	< *minsup*
milk, egg, diaper, beer	0 %	< *minsup*

Table 2.3: Confidence of large itemset for example 2.3

Rule	Confidence (α)	Rules Hold
milk ⇒ egg	= 0.4/0.8 = 50 %	No
milk ⇒ diaper	= 0.4/0.8 = 50 %	No
milk ⇒ beer	= 0.4/0.8 = 50 %	No
egg ⇒ diaper	= 0.4/0.8 = 50 %	No
egg ⇒ beer	= 0.5/0.8 = 63 %	Yes
milk, egg ⇒ beer	= 0.4/0.4 = 100 %	Yes

In general, rules mining algorithms are classified under two different approaches, namely, the level-wise approach [1, 2, 7-10] and the FP-tree growth approach [11, 12]. The well-known $Apriori$ algorithm uses the level-wise approach [1]. It scans the database multiple times to generate all frequent Itemsets. It first generates candidate Itemsets of size one, and calculates the support count for each single item. Then, It applies the $downward\ closure\ property$ to delete all infrequent itemsets. In the second round, Apriori generates candidate Itemsets of size two. It repeats the process until there is no candidate left. FP-tree growth improves the Apriori algorithm by reducing the number of database scans employed by Apriori [11]. FP-tree growth generates the frequent Itemsets with only two rounds over the database and without any candidate generation process. It first builds a data structure called FP-tree with two scans through the data. Then, it traverses through the tree to generate the frequent itemsets. FP-tree growth approach outperforms level-wise approach by only needing two rounds to scan the database.

Numerous algorithms are available in the literature for association rules mining that are either sequential [1, 7-12] or parallel [13-19]. In general, sequential algorithms are proposed based on the assumption that Itemsets are stored such that itemsets can be easily counted based on their lexicographical order [5]. Some key sequential algorithms are summarized in table 2.4. Other algorithms are extended versions of these algorithms.

Table 2.4: Summary of Sequential Algorithms

Algorithm	Authors	Overview and Strengths	Shortcomings
AIS [1]	Agrawal 93	• AIS is the first algorithm that has been introduced for mining frequent patterns. • It scans the database and counts and generates candidates, and determines the frequent patterns for each transaction. • It generates new candidates from those frequent patterns in the previous step.	• AIS requires multiple scans. • It unnecessarily counts and generates candidates that ultimately transpire to be small. • It needs space and time for un-useful candidates.
$Apriori$ [8]	Agrawal 94	• $Apriori$ is an optimization of AIS algorithm. • It adopts the same steps but it improves the time and space by introducing the $downward\ closure\ property$. • $Apriori$ outperforms AIS.	• $Apriori$ still requires multiple scans. • It still needs a lot of space and time.
$Partitioning$ [9]	Savasere 95	• It partitions data such that each partition can fit in main memory. • It generates the frequent patterns rules in only two rounds. • For each partition, it generates locally large itemsets using $Apriori$ in the first round. • It introduces the $property$ that states, a frequent itemset in the entire database must be frequent in at least one partition is. • It then combines the local large itemsets in each partition, and uses them as new candidates. • The new candidates are counted in the entire database to generate all the large itemsets. • It works well with a homogeneous data distribution.	• It generates a small candidate set when the data follow a skewed distribution.

Continuation of Table 2.4: Summary of Sequential Algorithms

Algorithm	Authors	Overview and Strengths	Shortcomings
$Sampling$ [10]	Toivonen 96	• It generates the frequent patterns rules in one round in the best case, and only two rounds in the worst case. • It first chooses a random sample that can fit in the main memory. • It generates the frequent patterns using $Apriori$ from this random sample instead of the entire database. • It introduces $Negative\ Border$ to generate more possible candidates, and use the rest of the database to verify the support of the candidates. • $Negative\ Border$ is a set that contains closest itemsets that could be frequent, and are not in the given frequent itemsets in the sample but have subsets of the frequent itemsets in the sample..	• Second scan to the database may be needed to generate another random sample In case of failure to find at least one frequent itemset in the negative border. • Thus, a large number of candidates might be found.
$FP-Tree$ [11]	Han 2000	• $FP-Tree$ is designed to handle the disadvantages that are associated with $Apriori$ algorithm such as the complexity and the multiple scans of the database. • It is faster than the $Apriori$ algorithm since it generates the frequent Itemsets in two rounds and does not require any candidate generation approach. • Unlike $Apriori$, it uses a compact data structure in order to avoid multiple scanning to generate candidates.	• $FP-Tree$ *consumes space.* • It is not suitable when the database is updated by adding new records or deleting existing ones.

On the other hand, the aim of parallel algorithms is to reduce the sequential algorithms' complexity by parallelizing the process of identifying large itemsets. Dunham et al. [5] classified most parallel or distributed association rules mining algorithms under two paradigms: data parallelism (DP) algorithms aim to parallelize data, and task parallelism (TP) algorithms aim to parallelize the candidates.

Data Parallelism Algorithms: Let C represent the set of candidates, and P represent the set of processors $\{P_1, P_2, \ldots\ldots, P_n\}$, such that C is duplicated on all processors. Let D represent the database $\{d_1, d_2, \ldots\ldots, d_m\}$ that is distributed over the processors. Let the set of support counts $\{s_1, s_2, \ldots\ldots, s_c\}$ represent the local support counts of all the candidates that are computed by the processors. Each processor is responsible for its own database. Let *Global support counts* represent the total support counts of the candidates in the entire database. The *Global support counts* of the candidates are computed in parallel for all the processors. This process is called *Global Reduction*, and can be done by swapping the local support counts. Finally, each processor independently computes the large Itemsets. Example 2.4 illustrates the concept of this task.

Example 2.4: Using the data in Table 2.6, the five transactions are partitioned across the three processors $(P1, P2, P3)$ as shown in table 2.8. Let $P1 = \{T1, T5\}$, $P2 = \{T2\}$, and $P3 = \{T3, T4\}$. The candidate Itemsets in the second scan are duplicated on each processor.

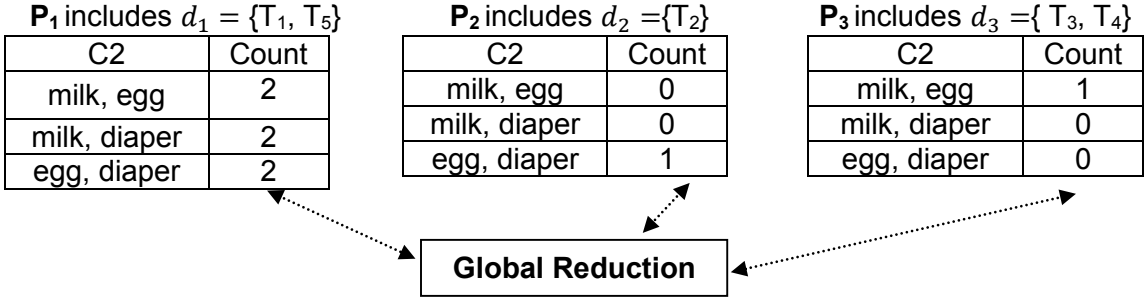
Table 2.6: Transactional Database for example 2.4

Transaction ID	**Items**
T_1	milk, egg, diaper
T_2	egg, diaper, beer
T_3	egg
T_4	milk, egg
T_5	milk, egg, diaper

Table 2.7: Support for the first and second Candidates for example 2.4

Itemset	**support (s)**	***minsup*=40%**
milk	60 %	> *minsup*
egg	100 %	> *minsup*
diaper	60 %	> *minsup*
beer	20 %	< *minsup*
milk, egg	60 %	> *minsup*
milk, diaper	40 %	> *minsup*
egg, diaper	40 %	> *minsup*

Table 2.8: Data parallelism paradigm

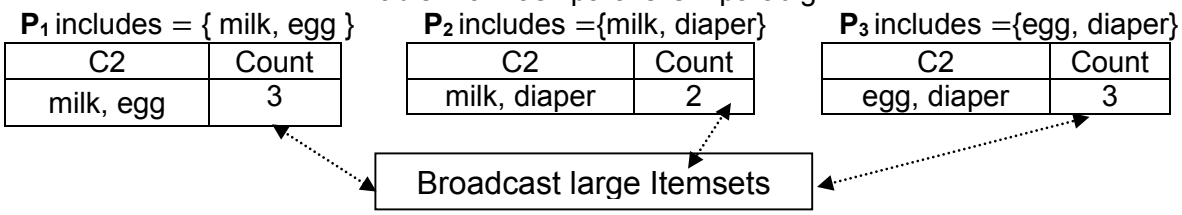


Task Parallelism Algorithms: Let C represent the set of candidates, let P represent the set of processors $\{P_1, P_2, \ldots\ldots, P_n\}$, and let D represent the database $\{d_1, d_2, \ldots\ldots, d_m\}$ such that C is partitioned into $\{c_1, c_2, \ldots\ldots, c_k\}$ and distributed across the processors $\{P_1, P_2, \ldots\ldots, P_n\}$ as is the database $\{d_1, d_2, \ldots\ldots, d_m\}$. In this paradigm, the *Global support counts* of the candidates are computed by each processor independently. This task can be done in only two rounds; firstly, each processor broadcasts its own candidates to the rest of the processors to calculate the large Itemsets. Then, after each processor calculates its own large Itemsets, they send their own large Itemsets to the rest

of the processors in order to compute the new candidates. Example 2.5 illustrates the concept of this task.

Example 2.5: Using the data in Table 2.7, the five transactions are partitioned across the three processors $(P1, P2, P3)$ as shown in table 2.9. Let us partition the candidate Itemsets across the processors with each processor having one candidate Itemset.

Table 2.9: Task parallelism paradigm



Count Distribution (CD) algorithm [13] uses the data parallelism (DP) paradigm, and follows the same steps as the algorithm above. Most of the parallel algorithms fall under DP and are extended versions of the CD algorithm. They are summarized in table 2.10.

Table 2.10: Summary of Parallel Algorithms under $the\ DP$ Paradigm

Algorithm	Authors	Overview and Strengths	Shortcomings
$Parallel\ Data\ Mining$ PDM Algorithm [14]	Park 95a	• PDM utilizes memory better than CD as direct hashing technique is added to prune some candidates in the second round.	• PDM needs to exchange not only the local counts of the candidate k-itemsets, but also the local counts in the hash table for k+1-itemsets.
$Common\ Candidate\ Partitioned\ Database\ CCPD\ Algorithm$ [15]	Zaki 96	• $CCPD$ generates and counts candidates in a shared-memory since it was proposed on a shared-memory system. • It uses the common prefixes (ex. the first item) to group large itemsets into equivalence classes, and then generates candidates from those classes. • To improve counting the candidates for each transaction, CCPD as well employs a short-circuited subset checking method.	• The common prefixes technique to group large itemsets helps in reducing the candidates generating time. However it does not reduce the number of generated candidates.

Continuation of Table 2.10: Summary of Parallel Algorithms under the DP Paradigm

Algorithm	Authors	Overview and Strengths	Shortcomings
$Distributed\ Mining\ DMA\ Algorithm$ [16]	Cheung 96	• DMA introduces two techniques; candidate pruning and communication message reduction. • Each processor calculates local counts of its own candidates to identify the heavy itemsets (Large itemsets that are large in both partition database locally, and in the whole database globally). • The candidates then are generated from the heavy large itemsets. • Unlike CD method that broadcasts local counts of all candidates, DMA shrinks the message size from $O\ (p^2)\ to\ O(p)$ by sending the local counts to only one polling site.	• $DMA's$ performance is sensitive to how the data is partitioned across the processors.

$Data\ Distribution\ (DD)$ algorithm [13] is follows the data parallelism (TP) paradigm, and follows the same steps of task parallelism algorithm above. Most of the parallel algorithms under TP are extended versions of the DD algorithm, and they are summarized in table 2.11.

Table 2.11: Summary of Parallel Algorithms under TP Paradigm

Algorithm	Authors	Overview and Strengths	Shortcomings
$Hash\ based\ Parallel\ Association\ mining\ of\ Association\ rules\ HPA\ Algorithm$ [17]	Shintani 96	• HPA uses a hash function to distribute, generate and count the candidates through processors. • Instead of allocating the database partitions in the processors, HPA only sends subset itemsets of the transactions. • It also uses skew handling technique, in order to achieve a load-balance for the candidates in each processor.	• It involves the maintenance of a Hash tree.

Continuation of Table 2.11: Summary of Parallel Algorithms under TP Paradigm

Algorithm	Authors	Overview and Strengths	Shortcomings
$Intelligent\ Data\ Distribution\ IDD\ Algorithm$ [18]	Han 97	• IDD distributes the candidates across the processors such that candidates share the first item, and share the same partition. • It eliminates a lot of redundant computations of DD. Each processor checks all subsets of each transaction, by only checking subsets that start with one items allocated to the processor.	• It uses complex structures to both partition candidates to reach to a load-balance on the candidate's distribution, and to minimize the overhead related to communication. • The increase of number of processors, decrease number of candidates set to each processor, which makes it difficult to achieve a load-balanced distribution, and reduce the efficiency.
$Parallel\ Association\ Rules\ PAR\ Algorithm$ [19]	Zaki 97	• PAR uses more than one methods for candidate partitioning and counting since it employs a set of algorithms (Par$Eclat$, $parMaxEclat$, $ParClique$ and $ParMaxClique$). • All PAR algorithms adopt a vertical database partition to accelerate the process of generating and counting the candidates.	• If the database is in horizontal layout, more steps will be needed to transform it to the vertical partition for all PAR algorithms.

Most of the existing works are designed under the horizontal layout. A horizontal layout keeps the data in transactions format where each transaction consists of different items (see table 2.12). A vertical layout is another layout for organizing the database [20-23]. A vertical layout views the data as a list of transactions, one for each item (see table 2.13). The list corresponding to any item is the list of transactions in which the item occurs.

Table 2.12: Horizontal Layout

TID	Items
T_1	milk, egg, diaper
T_2	milk, diaper
T_3	egg, beer
T_4	milk, egg, beer
T_5	milk, egg, beer

Table 2.13: Vertical Layout

TID	Items
milk	T_1, T_2, T_4, T_5
egg	T_1, T_3, T_4, T_5
diaper	T_1, T_2
beer	T_3, T_4, T_5

Since the database can be updated by adding new association rules, and invalidating some existing ones, many efforts have been introduced in the literature to come up with efficient algorithms to update, maintain, and manage the association rules [25-38]. These algorithms are mainly focused on handling the performance issues associated with re-scanning the original database when the database is updated (added or deleted); or when the thresholds are changed. There are four cases to consider when updating a database. Let the old transactional database be DB, and the new records (transactions) be db. Let the updated database then be $\acute{DB} = DB \cup db$. The four cases for the itemsets are summarized in table 2.14.

Table 2.14: The four cases of itemsets on an updated database

DB	db	
	Frequent Itemsets	**Infrequent Itemsets**
Frequent Itemsets	**Case 1:** itemset is large in both DB and db	**Case 2:** itemset is large only in DB
Infrequent Itemsets	**Case 3:** itemset is large only in db	**Case 4:** itemset is not large in both DB and db

Among the four cases, both *case 1* and *case 4* will change nothing in the existing association rules. However, *Case 2* might eliminate existing ones, while *case 3* might add new rules [39]. The main concept in incremental mining algorithms is, instead of rescanning the entire database again to discover the frequent itemsets, they reuse the information of frequent itemsets in the old database. So for *case 2*, the incremental mining algorithms do merge the itemsets' support of new db within the old counts in DB. The worst case in these algorithms lies in *case 3* since the only solution is to rescan the entire database. To handle *case 3,* Hong et al. [31] proposed a novel algorithm, and introduced the notion of a pre-large itemset. A pre-large itemset is not a frequent (large) itemset, but it is capable of becoming large in the updated database. For this, they introduced two support thresholds; one stands for identifying the frequent itemsets like the other

traditional algorithms. The second threshold stands for identifying the pre-large itemsets. Based on this solution, an itemset may have the nine cases summarized in table 2.15.

Table 2.15: The nine cases of itemsets in an updated database

DB	db		
	Frequent Itemsets	**Pre-large Itemsets**	**Infrequent Itemsets**
Frequent Itemsets	**Case 1:** itemset is large in both DB and db **Result:** itemset will definitely be frequent	**Case 2:** itemset is large in DB and pre-large in db **Result:** itemset might be frequent or pre-large	**Case 3:** itemset is large only $in\ DB$ **Result:** itemset might be frequent or pre-large, or infrequent
Pre-large Itemsets	**Case 4:** itemset is pre-large in DB and large in db **Result:** itemset might be pre-large or a frequent	**Case 5:** itemset is pre-large in DB and db **Result:** itemset will definitely be pre-large	**Case 6:** itemset is pre-large in DB and small in db **Result:** itemset might be pre-large or infrequent
Infrequent Itemsets	**Case 7:** itemset is large only in db **Result:** itemset might be pre-large or infrequent	**Case 8:** itemset is not large in DB and pre-large in db **Result:** itemset might be infrequent or pre-large	**Case 9:** itemset is not large in both DB and db **Result:** itemset will definitely be infrequent

Among the nine cases, both *case 1* and *case 9* will change nothing in the existing association rules. Also, *Case 2, Case 3,* and *Case 4* can be handled effectively, and can be determined by retaining all frequent and pre-large itemsets with their supports in the old database. For *case 7*, it is unlikely for an itemset to be large in the updated database since the fraction of the number db over the number of DB is usually trivial.

2.2 Association Rules Mining on Uncertain Data

Traditional association rules maiming algorithms assume a transactional database where for each transaction we know for sure the items that belong to the transaction. In

many real life applications, the database is uncertain, and for any transaction, we only know a probability for each possible item that this item belongs to the transaction [40]. In this case the frequency of an item (or an itemset) is calculated as the expected number of occurrences of the item (itemset) in the transactions [41, 39]. Research has also been conducted to mine rules from uncertain data [40-46] due to the growth of applications that involve uncertain data such as data from social networks, sensor networks [47], protein-protein interaction analysis [48], and inaccurate surveys. Uncertainty could arise from masking of data for privacy concerns as well [49].

Definition 2.4: The problem of mining from uncertain data can be formulated as follows. Let the set of distinct items be $I = \{i_1, i_2, \dots, i_m\}$. Consider an uncertain database UDB which contains a set of N transactions, $T = \{t_1, t_2, \dots, t_N\}$.. Here the transaction t_j is characterized as a probability vector $p_j(i_1), p_j(i_2), \cdots, p_j(i_m)$ where $p_j(x)$ is the probability that the transaction t_j contains the item x, for any item x and $1 \leq j \leq N$. Let X be any itemset. We can define the expected support of X as follows.

Definition 2.5: Let UDB be an uncertain database with N transactions, and let X be any itemset. Then, the expected support of X is given by the following equation: $esup(X) = \sum_{j=1}^{N} \prod_{x \in X} p_j(x)$.

Definition 2.6: An itemset X is said to be frequent if and only if its expected support $esup(X)$ is atleast $(N \times minsup)$ where N is the number of transactions and $minsup$ is the minimum expected support required and it is specified by the user.

Example 2.6: Table 2.16 shows a small UDB where each row represents one transaction. If t is any transaction, then all the items in this transaction that have a nonzero probability of occurrence are listed in column 2 of the corresponding row. Since the itemset $\{a, b\}$ appears only in one transaction t_1, then $esup(\{a, b\}) = [0.1033 \times 0.865] = 0.08935$. Also, since the itemset $\{b, d\}$ appears together in t_1, t_2 and t_3, then $esup(\{b, d\}) = [(0.865 \times 0.726) + (0.906 \times 0.726) + (0.882 \times 0.897)] = 2.0769$. If $minsup$ is given as 0.3, then the itemset $\{b, d\}$ qualifies as a frequent itemset (based on the expected support) whereas the itemset $\{a, b\}$ does not qualify.

Table 2.16: An uncertain database

TID	Transactions
t_1	a (0.1033) b (0.865) c (0.919) d (0.726)
t_2	c (0.854) b (0.906) d (0.726)
t_3	b(0.882) e(0.853) d(0.897)

2.2.1 Frequent patterns Algorithms for Uncertain Data

Most of the frequent patterns algorithms under uncertainty, are extended versions of those traditional rules mining algorithms with certain data. For example, UApriori which extends the traditional Apriori algorithm is first algorithm introduced for finding frequent itemsets based on expected support in uncertain databases. Chui, et al. [45] have introduced the UApriori algorithm that is based on the computation of expected supports. Like the traditional Apriori, it adapts a support-based frequent itemsets recursively. In uncertain databases, downward closure property [8] also is satisfied. So, we still can prune all the supersets of expected support-based infrequent itemsets. Chui, et al. [45] have also proposed decremental pruning methods to improve the efficiency of UApriori. The decremental pruning methods are employed to estimate an upper bound on the

expected support of an itemset from the beginning, and the traditional Apriori pruning is used when the upper bound is lower than the minimum expected support. Yet, the traditional Apriori pruning is mainly used in UApriori since the decremental pruning methods depend on the structure of the datasets.

UFP-growth [46] is similar to the certain FP-growth algorithm. It first builds a compact data structure called the UFP-tree using only two passes over the uncertain database. Then, it constructs conditional sub-trees and finds expected support-based frequent itemsets from the UFP-tree. Unlike FP-growth, where the compact FP-tree shrinks different transactions that share common subsets/prefixes in only one path, UFP-tree is substantially reduced. Thus, items may share only one node when both their ids and probabilities are common. Otherwise, even common subsets/prefixes that don't share the same probabilities will be represented in two different nodes/paths. Due to fewer shared nodes and paths, an uncertain database can be viewed as a sparse database. A lot of redundant computations take place while processing a UFP-tree since the number of conditional sub-trees could be very large. In fact, the performance of the deterministic FP-growth cannot be achieved using UFP-growth.

Leung, et al. [47] have suggested a straightforward solution by considering each distinct probability as a different item. This solution can be effective only when many items have the same probability which is not the typical case in the domain of uncertain databases [41]. Agarwal, et al. [41] suggested another solution by creating clustered ranges of probabilities and for each clustered range the relevant nodes are linked. Followed by this, FP-Tree algorithm is used to generate a close superset of the frequent

itemsets. They also suggested another solution by using an extended H-Mine, namely, U H-Mine [41].

Unlike the traditional Apriori algorithm in deterministic databases, the performance of UApriori is better than the other mining algorithms in the domain of uncertain data. It is considered as one of the fastest for dense uncertain datasets in general [41].

Chapter 3

Frequent Itemsets Mining From Weighted Uncertain Data

3.1 Introduction

Mining frequent itemsets from datasets is a well-studied problem. Traditional approaches to mining frequent patterns have suffered from the fact that all the items are treated as equally important. In real applications, each item has a different level of importance. For example, in retail applications some products may be much more expensive than the others, and these expensive items may not be present in a large number of transactions. The importance of weight-based pattern mining approach can be felt in many domains such as, biomedical data analysis where the causes of most diseases are not only one gene but a combination of genes; web traversal pattern mining where the impact of each web page is different; and so on. Several variations of this problem have also been investigated in the literature [50-54]. On the other hand, there are many applications where the data are both weighted and uncertain. To the best of our knowledge, mining from such datasets has not been studied before. In this research work

we initiate the study of frequent itemsets mining from weighted uncertain data. In particular, we propose two algorithms called $HWUAPRIORI$ and $VWUFIM$ for mining frequent itemsets from weighted uncertain data [55]. We evaluate the performance of the proposed algorithms on various datasets.

Definition 3.1: Let $w(I)$ be a positive real number that stands for the weight of an item. It represents the importance of the item in the transaction database. Then, the weight of an itemset X is the average weight of items, and is defend as:

$$W(X) = \frac{\sum_{1}^{length\,(X)} w_X}{length\,(X)}.$$

Definition 3.2: Weighted support of an itemset is defined as:

$$Wsupport(X) = W(X) \times Support\,(X).$$

Definition 3.3: An itemset X is said to be a weighted frequent pattern if

$$Wsupport(X) \geq minsup.$$

Many algorithms have been proposed for weighting items based on their significance. Unfortunately, the $downward\ closure\ property$ does not hold for weighted data making it a challenge to develop mining algorithms [1].

Example 3.1: Table 3.1 shows a small certain data, where each transaction has a subset of items, and each item is assigned with weight in table 3.2. Assume that the minimum threshold, $minsup = 0.5$. Weighted support for pattern diaper is

$Wsupport(\text{diaper}) = 0.6 \times \frac{3}{4} = 0.45 < minsup$, while weighted support for each pattern (egg, diaper) is $Wsupport(egg, \text{diaper}) = \frac{(0.5 + 0.6)}{2} \times \frac{4}{4} = 0.55 > minsup$. However, if we follow the rule of $downward\ closure\ property$ we will delete item $(diaper)$ in the early stage, and we will lose the itemset $(egg, diaper)$ that has a weighted support greater than the $minsup$. As we can see using weights violates the well-known $downward\ closure\ property$ that has a great impact on reducing search space, and time.

Table 3.1: Certain data for example 3.1

TID	Transactions
T_1	milk, egg, diaper, beer
T_2	egg, diaper, beer
T_3	egg, diaper, beer, chees
T_4	egg, diaper

Table 3.2: Items' weights for example 3.1

Items	Weight
milk	0.6
egg	0.5
diaper	0.6
beer	0.5
chees	0.1

3.2 Weighted frequent pattern mining algorithms on certain database

Both WARM [51], and WAR [52] were proposed based on the Apriori algorithm which is a level-wise approach to generate weighted association rules. The authors of WARM [51] introduced two varieties of weight; a weight assigned for each individual item, and an itemset weight. The itemset weight is the average weight of the items. An interesting rule has to satisfy the support and confidence thresholds, and the weighted support has to be at least a user-defined threshold weighted minimum support called $wminsup$. The

weighted support is the fraction of the transactions' weight (that contains itemset) relative to the total transactions' weight.

Example 3.2: using items' weight in table 3.2 each individual item is assigned with weight. Table 3.3 shows a small certain data, where the weight for each transaction is calculated in the column $TWeight$, (i.e. weight of transaction $T_1 = \frac{0.6+0.5+0.6+0.5+0.1}{5} = 0.46$). Assume that both minimum threshold $minsup = 0.5$, and weighted minimum threshold $wminsup = 0.5$. Support for pattern (egg, diaper) is $sup(egg, \text{diaper}) = \frac{3}{4} = 0.75 > minsup$, and weighted support for pattern (egg, diaper) is $Wsupport(egg, \text{diaper}) = \frac{0.46+ 0.53+ 0.425}{2.015} = 0.70 > wminsup$

Table 3.3: Weighed transactional certain data for example 3.2

TID	Transactions	TWeight
T_1	milk, egg, diaper, beer, chees	0.46
T_2	egg, diaper, beer	0.53
T_3	egg, diaper, beer, chees	0.425
T_4	diaper	0.6
Total transaction weight		2.015

Giving a weight range for each individual item, WAR [52] starts by generating the frequent itemsets using the traditional Apriori algorithm without considering item/itemset weight. It then, generates the weighted association rules under three conditions; weighted association rule has to satisfy both support and confidence thresholds, and new measurement called density threshold using space partition. Yun, et al. [53] have proposed $WFIM$ as the first algorithm based on FP-tree growth algorithm for frequent itemset mining from weighted data. Similar to FP-tree growth algorithm, $WFIM$ scans the database twice. In order to ensure the $downward\ closure\ property$, a minimum weight and a weight range

are used, where each item has a random fixed weight W in a weight range: $W_{min} \leq W \leq W_{max}$. An itemset X is a weighted infrequent itemset if, following pruning, condition 1 **or** condition 2 below is satisfied:

1. Pruning condition 1: $sup(X) < minsup \ \&\& \ W(X) < W_{min}$
2. Pruning condition 2: $Wsup(X) < minsup$, where $Wsup(X) = sup(X) \times W_{min}$

Here $sup(X)$ stands for the support of X, $minsup$ is the minimum support threshold, $W(X)$ is the average weight of the items in X, W_{min} is the minimum weight, and $Wsup(X)$ is weighted support.

Yun [54] has proposed an algorithm called WIP based on the FP-tree growth algorithm. He has also defined the concept of a weighted $hyperclique$. This is a new measure of weight-confidence that measures the weight affinity of a pattern and prevents the generation of patterns that have significantly different weight levels. Weight confidence of a pattern $X = \{I_1, I_2, I_3, \dots, I_m\}$ is defined as the ratio of the minimum weight of items to the maximum weight of items:

$$Wconf(X) = \frac{Min_{\,1 \leq i \leq m} \{\text{weight}(\{\text{I}_\text{i}\})\}}{Max_{\,1 \leq j \leq m} \{\text{weight}(\{\text{I}_\text{j}\})\}}.$$

If the weight confidence of a pattern is greater than or equal to a minimum weight confidence, then the pattern is called a weighted $hyperclique$ pattern. An itemset X is a weighted infrequent itemset if, following pruning, condition 1 **or** condition 2 **or** condition 3 below is satisfied:

1. Pruning condition 1: $sup(X) \times W_{min} < minsup$
2. Pruning condition 2: $Wconf(X) < \min\_wconf$.

3. Pruning condition 3: $h_{confidence} < min_{hconf}$, where, the $h - confidence$ of a pattern $X = \{I_1, I_2, I_3, \dots, I_m\}$ is defined as: $\frac{\text{support } (\{I_1, I_2, I_3, \dots, I_m\})}{Max\ _{1 \le j \le m} \{support\ (\{I_j\})\}}$.

3.3 Proposed Algorithms of Weighted frequent pattern mining algorithms on uncertain data

In this section we present elegant algorithms for mining weighted frequent itemsets from uncertain databases. In the $WFIM$ algorithm [53], for the case of certain data, an itemset X is a weighted infrequent itemset if, following pruning, condition 1 **or** condition 2 below is satisfied:

1. Pruning condition 1: $sup(X) < minsup\ \&\&\ W(X) < W_{min}$
2. Pruning condition 2: $Wsup(X) < minsup$, where $Wsup(X) = sup(X) \times W_{min}$

Therefore, in uncertain data, we can say, an itemset is frequent if both following pruning conditions are not satisfied:

1. Pruning condition 1: $esup(X) < min1_{esup}\ \&\&\ W(X) < W_{min}$
2. Pruning condition 2: $Wesup(p)\ [= esup(X) \times W(X)] < min2_{esup}$

Where $esup$ is the expected support, and $Wesup$ is the weighted expected support. We introduce two support parameters $min1_{esup}$ and $min2_{esup}$, where $min1_{esup}$ stands for the first pruning condition, and $min2_{esup}$stands for the second pruning condition. Also, $min2_{esup}$ should be less than $min1_{esup}$. Our algorithms for weighted uncertain frequent itemset mining are designed using two layouts namely $horizontal$ and $vertical$. $Horizontal\ Layout$ of any dataset keeps the data as transactions where each transaction is an itemset. On the other hand, the $vertical\ layout$ keeps the data as a list of

transactions for each item. The list associated with any item is the list of transactions in which the item is present. Below we present the horizontal weighted uncertain Apriori algorithm.

Example 3.3: Consider the uncertain transaction database UDB shown in Table 3.4. Table 3.5 shows each individual item is assigned with weight. Note just for testing, we give some item that has low support, high weight and via versa. Let $minsup1 = 0.5$, $W_{min} = 0.1$, and number of tranactions $N = 3$.

Table 3.4: uncertain transaction database for example 3.3

TID	Transactions
T_1	1 0.933 2 0.865 3 0.919 4 0.726
T_2	2 0.854 3 0.906 4 0.726
T_3	3 0.933 4 0.865 5 0.919

Table 3.5: Item's weight for example 3.3

Items	Weight
1	0.9
2	0.6
3	0.2
4	0.5
5	0.1

$min1_{esup} = N \times minsup1 = (3 \times 0.5) = 1.5$, and $min2_{esup} = \frac{min1_{esup}}{N} = \left(\frac{1.5}{3}\right) = 0.5$

- $esup\,(1) = 0.9 < min1_{esup}$
- $esup\,(2) = 0.865 + 0.854 = 1.7 > min1_{esup}$
- $esup\,(3) = 0.919 + 0.906 + 0.933 = 2.758 > min1_{esup}$
- $esup\,(4) = 0.726 + 0.726 + 0.865 = 2.317 > min1_{esup}$
- $esup\,(5) = 0.919 < min1_{esup}$

According to Pruning condition 1, $esup\,(5) = 0.919 < min1_{esup}$ **&&** $W(5) < W_{min}$. Since $item\ (5)\ infrequnt$, then, we don't need to test item (5) on pruning condition 2.

According to Pruning condition 2,

- $Wseup(1) = 0.9 \times 0.933 = 0.8 > min2_{esup}$
- $Wseup(2) = 0.6 \times (0.865 + 0.854) = 0.6 \times 1.719 = 1.03 > min2_{esup}$
- $Wseup(3) = 0.2 \times (0.919 + 0.906 + 0.933) = 0.2 \times 2.758 = 0.5516 > min2_{esup}$
- $Wseup(4) = 0.5 \times (0.726 + 0.726 + 0.865) = 0.5 \times 2.317 = 1.1585 > min2_{esup}$

Let return item (5) to test the downward closure property:

- $W(1,2) = \left[\frac{0.9 + 0.6}{2}\right] = 0.75 > W_{min}$
- $W(1,3) = \left[\frac{0.9 + 0.2}{2}\right] = 0.5 > W_{min}$
- $W(1,4) = \left[\frac{0.9 + 0.5}{2}\right] = 0.7 > W_{min}$
- $W(2,3) = \left[\frac{0.6 + 0.3}{2}\right] = 0.45 > W_{min}$
- $W(2,4) = \left[\frac{0.6 + 0.5}{2}\right] = 0.55 > W_{min}$
- $W(3,4) = \left[\frac{0.2 + 0.5}{2}\right] = 0.35 > W_{min}$
- $W(3,5) = \left[\frac{0.2 + 0.1}{2}\right] = 0.15 > W_{min}$
- $W(4,5) = \left[\frac{0.5 + 0.1}{2}\right] = 0.3 > W_{min}$

And:

- $esup\ (1,2) = 0.8 < min1_{esup}$ *But,* $W(1,2) > W_{min}$ ➔ $esup\ (1,2)\ frequnt$
- $esup\ (1,3) = 0.9 < min1_{esup}$ *But,* $W(1,3) > W_{min}$ ➔ $esup\ (1,3)\ frequnt$
- $esup\ (1,4) = 0.7 < min1_{esup}$ *But,* $W(1,4) > W_{min}$ ➔ $esup\ (1,4)\ frequnt$
- $esup\ (2,3) = 1.6 > min1_{esup}$ *And,* $W(2,3) > W_{min}$ ➔ $esup\ (2,3)\ frequnt$
- $esup\ (2,4) = 1.2 < min1_{esup}$ *But,* $W(2,4) > W_{min}$ ➔ $esup\ (2,3)\ frequnt$
- $esup\ (3,4) = 2.1 > min1_{esup}$ *And,* $W(3,4) > W_{min}$ ➔ $esup\ (3,4)\ frequnt$
- $esup\ (3,5) = 0.9 < min1_{esup}$ *And* $W(3,5) > W_{min}$ ➔ $esup\ (3,5)\ frequnt$

- $esup\ (4,5) = \ 0.8 \ < \ Min\_esup$ *And* $W(4,5) > \ W_{min}$ ➔ $esup\ (4,5)\ frequnt$

According to Pruning condition 2:

- $Wseup(1,2) = \ 0.75 \ \times [0.933 \ \times \ 0.865] \ = \ 0.75 \ \times 0.8 = 0.6 > \ min2_{esup}$
- $Wseup(1,3) = 0.55 \times \ [0.933 \ \times \ 0.919] = 0.55 \ \times \ 0.9 \ ] \ = \ 0.5 = \ min2_{esup}$
- $Wseup(1,4) = 0.7 \times [0.933 \ \times \ 0.726] = \ 0.7 \ \times \ 0.7 = \ 0.5 = \ min2_{esup}$
- $Wseup(2,3) = 0.45 \ \times [(0.865 \ \times \ 0.919) \ + \ (0.854 \ \times \ 0.906] = 0.45 \ \times 1.6 = \ 0.7 \ > \ min2_{esup}$
- $Wseup(2,4) = 0.55 \ \times [(0.865 \ \times \ 0.726) + \ (0.854 \ \times \ 0.726)] = 0.55 \ \times 1.2 = \ 0.7 > \ min2_{esup}$
- $Wseup(3,4) = 0.35 \ \times [(0.919 \ \times \ 0.726) + (0.906 \ \times \ 0.726) + (0.933 \ \times 0.865)] = 0.35 \ \times 2.1 = 0.7 > min2_{esup}$
- $Wseup(3,5) = 0.15 \ \times \ [0.933 \ \times \ 0.919] = \ 0.15 \ \times 0.9 = \ 0.1 < \ min2_{esup}$
- $Wseup(4,5) = \ 0.3 \times [0.865 \ \times \ 0.919] = \ 0.3 \times 0.8 = \ 0.2 < \ min2_{esup}$

  ➔ *We burn* $(3,5)$, $(4,5)$ *since* $esup\ (3,5), (4,5) < \ min2_{esup}$

3.3.1 Horizontal Weighted Uncertain Apriori $(HWUAPRIORI)$

Many frequent itemset mining algorithms employ the following strategy: 1-frequent itemsets are generated first; followed by this candidate 2-frequent itemsets are generated; for each candidate 2-frequent itemset, its support is calculated to check if it is indeed frequent; after this we have determined all the 2-frequent itemsets; followed by this we generate candidate 3-frequent itemsets; and so on. Several techniques have been introduced to reduce the computational complexity for generating the candidate itemsets. An example is the $F^{k-1} \ \times \ F^{k-1}$ method. Efficient data structures have been introduced to reduce the number of comparisons. Examples include hash tables, FP-trees, etc. [24].

In the $F^{k-1} \times F^{k-1}$ method, a pair of frequent (k-1)-itemsets are merged to get a candidate frequent k-itemset if and only if their first k−2 items are equal.

Let $A = \{a_1, a_2\ a_3, \ldots\ldots a_{k-1}\}$, and $B = \{b_1, b_2\ b_3, \ldots\ldots b_{k-1}\}$ be two frequent itemsets. Then A and B are merged to get a candidate frequent k-itemset if and only if: $a_{i,=} b_i\ for\ all\ i = 1, \ldots\ldots k-1$.

Input:

- *A transaction set T with $|T| = N$,*
- *A set I of items in the transactions,*
- *$minsup1,\ minsup2$,*
- *Weights of the items $w_1, w_2, \ldots, w_m$,*
- *Minimum weight threshold: min_weight.*

Output:

All the Uncertain Weighted Frequent Itemsets.

Algorithm:

Step 1: Pre-Processing

$min1_{esup} = \ minsup1 \times N$, *and* $min2_{esup} = \ \frac{min1_{esup}}{N}$

Step 2: /* Find the uncertain weighted frequent items with $(\boldsymbol{esup}(x) \geq \boldsymbol{min1_{esup}}\ ||\ \boldsymbol{W}(x) \geq \boldsymbol{min\_weight})\ \&\&\ (\boldsymbol{esup}(x) \times \boldsymbol{W}(x) \geq \boldsymbol{min2_{esup}})$****/***

1. *Scan through the database to find weighted frequent items x that do not satisfy both pruning conditions.*

Condition 1: $(esup(x) < min1_{esup}\ \&\&\ W(x) < min\_weight)$

Condition 2: $(esup(x) \times W(x) < min2_{esup})$

2. *Update the database by deleting all infrequent items that satisfy at least one of the pruning conditions.*
3. *Scan through the database to find uncertain weighted candidate 2-itemsets x that do not satisfy both pruning conditions*

4. Delete all candidate 2-itemsets that satisfy at least one of the pruning conditions.

5. For all candidate itemsets of size greater than two do:

5.1. Apply $F^{k-1} \times F^{k-1}$ method to generate the candidate itemsets.

5.2. Next, the database is scanned again for matching each transaction against candidates contained in the hashed buckets.

5.3. The support of candidates are counted, and the two pruning conditions are checked.

5.4. A candidate is a weighted frequent itemset if does not satisfy both two conditions.

5.5. Prune the weighted infrequent k-itemsets.

6. Generate all the Weighted Frequent Itemsets.

The horizontal $UWApriori$ algorithm has some drawbacks. First, it merges two frequent (*k*-1)-itemsets with (*k*-2) items common between them to come up with a candidate k-itemset. This requires $O(k)$ time given that the items are all ordered. Also, the algorithm has to traverse through the entire database of transactions to figure out if a *k*-itemset is frequent. Even for very dense databases a typical itemset is present in only a fraction of all the transactions. Checking the entire database is an overkill. These issues are addressed in the Vertical Weighted Uncertain Frequent Itemset Mining ($VWUFIM$) algorithm. Below we provide an overview of the $VWUFIM$ algorithm.

3.3.2 Vertical Weighted Uncertain Frequent itemset Mining Algorithm $VWUFIM$

This algorithm works on a different layout than its $horizontal$ counterpart. We first order the items and transactions. The database is changed in the following form : $I_j \rightarrow T(I_j)$, where I_j is the j^{th} item and $T(I_j)$ is a list of transactions that contain I_j We can convert the

horizontal layout to vertical layout in just a single scan. A vertical layout removes the shortcomings of the horizontal $UWApriori$ algorithm in the following way. First, we generate a *k*-frequent itemset by merging a (*k*-1)-frequent itemset with a 1-frequent itemset in ascending order of item IDs. Note that, this can be in $O(k)$ time. Second, to compute the support for a *k*-itemset, we perform intersection between transaction lists. Due to ordering of transactions this intersection has a time complexity of linear in the total length of the two lists. Below we provide the pseudo code for $VUWFIM$ algorithm.

Input:

- *A database of all transactions.*
- *Total number of transactions,*
- *minsup1,*
- *minsup2,*
- *Weights of the items,*
- *Minimum weight threshold:* min_weight.

Output:

All the Uncertain Weighted Frequent Itemsets.

Algorithm:

1. *If the dataset is in horizontal format, convert it to vertical format. A vertical format is given by a set of items each associated with a set of transactions the item appears in.*
2. *Find out the set of 1-frequent itemsets in a single pass through the data. We use the same conditions as mentioned in algorithm HWUAPRIORI.*
3. *Generate k-frequent itemsets.*
   3.1. Generate candidate k-itemsets using the $F^{k-1} \times F^1$ *method.*

3.2. If A is a list of transactions that a frequent (k-1)-itemset x occurs in and B is a list of transactions in which a frequent item y occurs in, then the support for the candidate k-itemset z obtained from x and y can be computed by intersecting A and B. Apply the pruning conditions to check if z is frequent.

4. Repeat step3 until all the frequent itemsets are found.

3.4 Complexity Analysis:

- Let n be the number of transactions
- q = average number of items in a transaction
- m = total number of items
- ρ = average size of each item list
- I^k= a generic *k*-itemset
- $T(I^k)$ = a list of transactions that I^k appears in.
- F^k = set of all *k*-frequent itemsets

3.4.1 Time Complexity Analysis for HWUAPRIORI:

$HWUAPRIORI$ requires $O(m)$ as a single scan through the set of all 1-itemsets for the first step. It requires $O(F^{k-1})| * (k) * |F^1)$ total time to generate set of all the candidate *k*-itemsets I^k from F^{k-1} and F^1. To check whether any I^k is frequent or not we have to scan through the dataset each time. Hence the total average time is given by $[m + \sum_{j=1}^{(k'-1)} \left|F^{(j)}\right| |F^{(1)}| jqn]$, where $k' - 1$ is the number of items in the largest frequent itemset.

3.4.2 Time Complexity Analysis for VWUFIM:

In $VWUFIM$ algorithm step 1 takes $O(q * n)$ time. Each item is allocated a bucket indexed with the same id as the item. In single scan through the database we can figure out $T(I^1)$ for all 1-itemsets. Step 2 takes a single scan through the set of all 1-itemsets. Time complexity for this step is $O(m)$. Step 3 is most crucial in the algorithm because this step is iterated. Step 3.1 takes $O(k)$ time for a (k-1)-frequent itemset. Total time for all the k-1 itemsets in the set is given by $O(|F^{(k-1)}| * (k-1) * |F^{(1)}|)$. Step 3.2 requires an expected time of $O\left(\binom{m}{k} p\right)$. Hence the total runtime for algorithm $VWFIM$ is given by $O\left(m + \binom{m}{k} p\right)$. Hence $VWUFIM$ is faster than $HWUAPRIORI$ in practice. Although the ratio of runtime is expected to vary based on other parameters provided by the users. In the next section, we show tables that display experimental results supporting our theoretical analysis.

3.5 Experimental Results

In this section, we present some experimental results on $HWUAPRIORI$ and $VWUFIM$.

$HWUAPRIORI$ and $VWUFIM$ are implemented and compiled using Microsoft's Visual Studio C++ 2013 and Java respectively. Following is the execution environment:

- Intel(R) Core(TM) i7 3.40GHz PC
- 8GB Main memory
- Operating System is Microsoft Windows 7.

For testing the performance of $HWUAPRIORI$ and $VWUFIM$over weighted uncertain databases, we have used:

A sparse dataset consisting of 9982 transactions with 336 different items, and the size of largest itemset is 267 items. Dense datasets have 10,000 transactions with 992 different items, and the size of the largest itemset is 267 items.

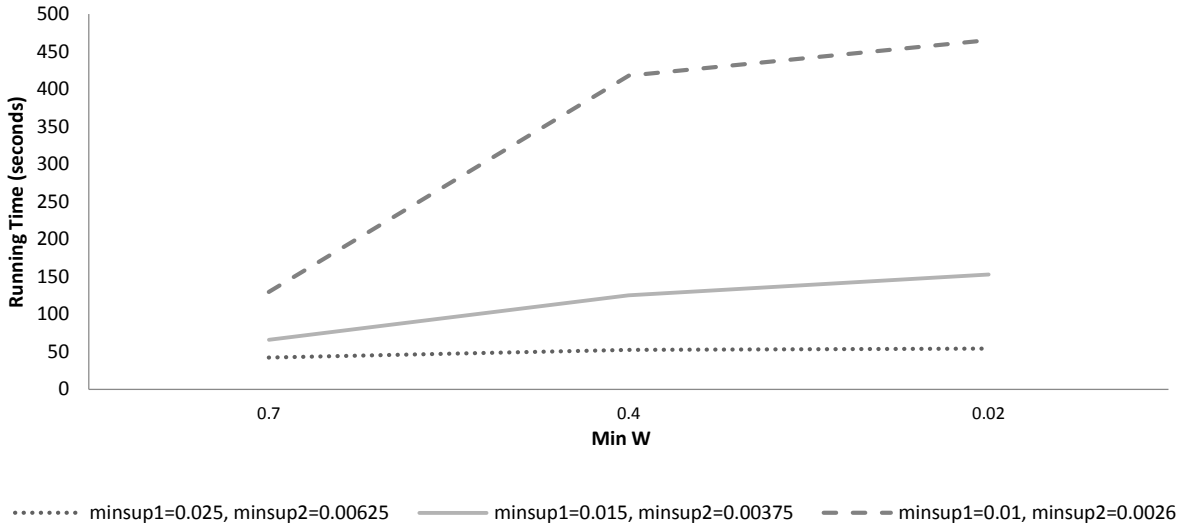
For each dataset, we have run our algorithms with different values of minimum weights, and different configuration parameters: $minsup1$, and $minsup2$.

Figure 3.1 shows the performance of our proposed algorithms and figure 3.2 shows a comparison between the algorithms VWUFIM and HWUAPRIORI in terms of run times. **Note** that the runtimes can vary based on other parameters provided by the users.

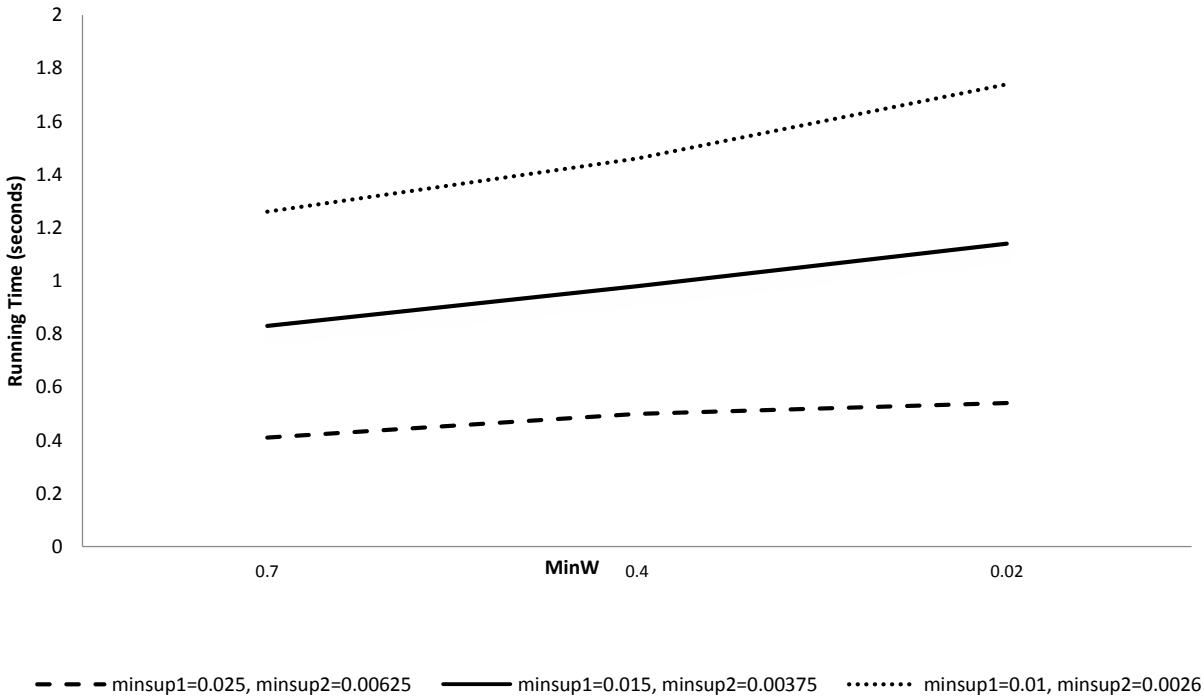
3.6 Conclusion

Many real life problems can be addressed using uncertain association rules mining. Many of these problems come with items having different weight factors. In this research work we have introduced and studied the problem of mining from uncertain weighted data. To the best of our knowledge we are the first ones to introduce this problem. We have also proposed two algorithms to tackle this problem. Our algorithm $HWUAPRIORI$ works on weighted uncertain horizontal databases while $VWUFIM$ works on vertical layout for the same problem. We have presented theoretical and experimental results for the proposed methods.

(a) HWUAPRIORI:
Sparse dataset: 9982 Transactions with 336 Items. Largest transaction size=267 itemset



(b) VWUFIM
Sparse dataset: 9982 Transactions with 336 Items. Largest transaction size=267 itemset



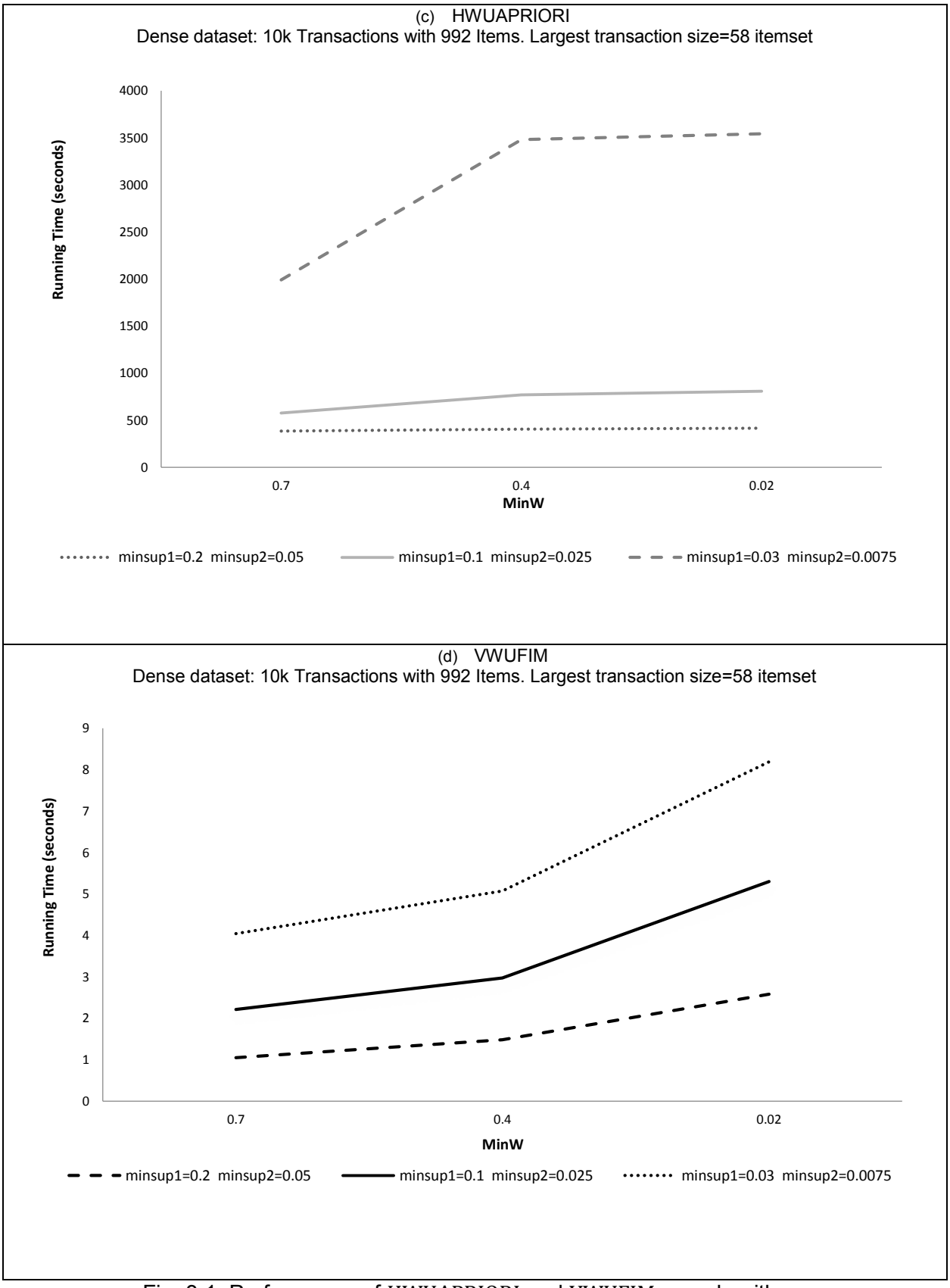
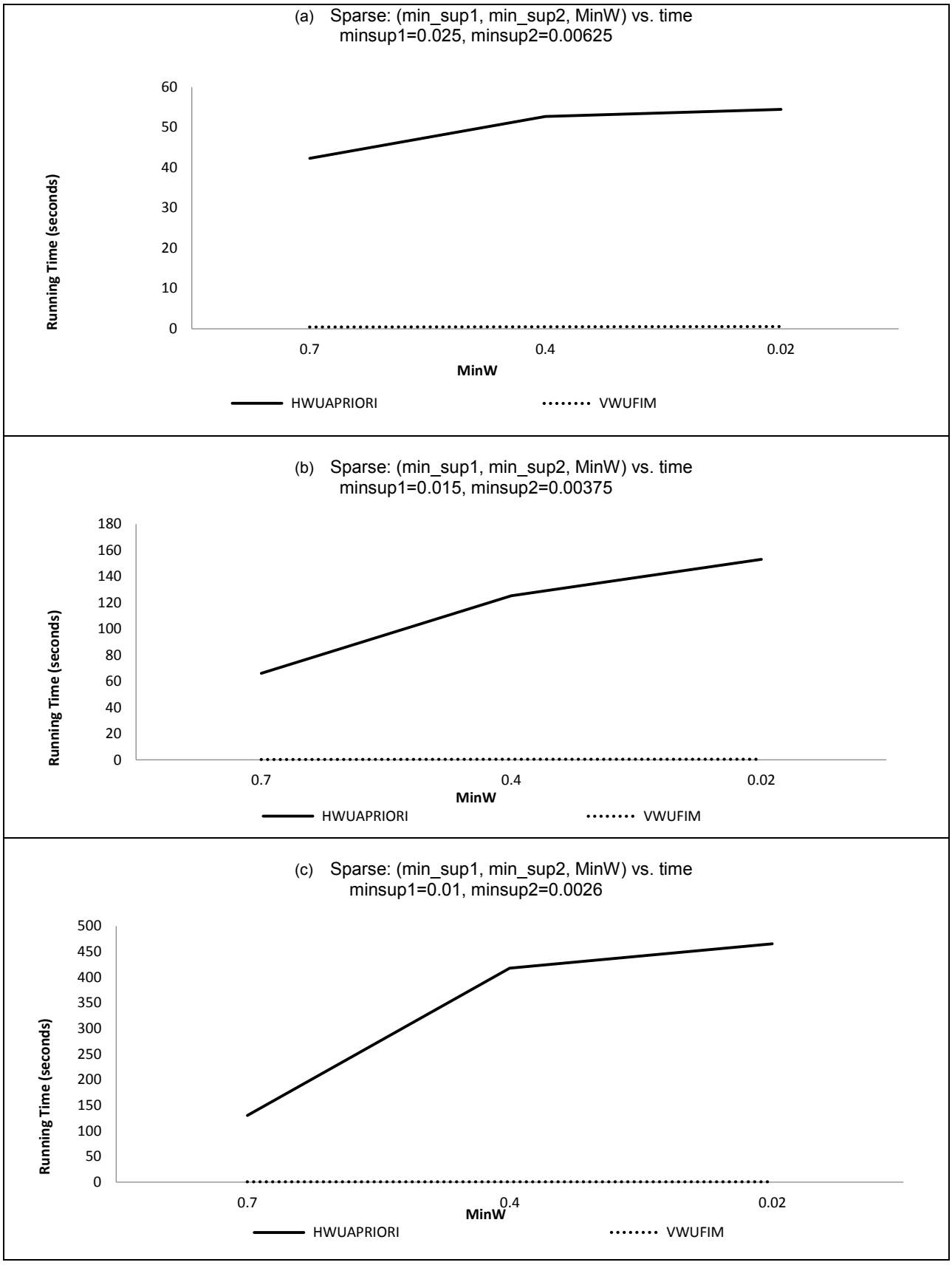


Fig. 3.1. Performance of HWUAPRIORI and VWUFIMover algorithms


(a) Sparse: (min_sup1, min_sup2, MinW) vs. time
minsup1=0.025, minsup2=0.00625
Running Time (seconds)
60
50
40
30
20
10
0
0.7
0.4
0.02
MinW
HWUAPRIORI
VWUFIM
(b) Sparse: (min_sup1, min_sup2, MinW) vs. time
minsup1=0.015, minsup2=0.00375
Running Time (seconds)
180
160
140
120
100
80
60
40
20
0
0.7
0.4
0.02
MinW
HWUAPRIORI
VWUFIM
(c) Sparse: (min_sup1, min_sup2, MinW) vs. time
minsup1=0.01, minsup2=0.0026
Running Time (seconds)
500
450
400
350
300
250
200
150
100
50
0
0.7
0.4
0.02
MinW
HWUAPRIORI
VWUFIM

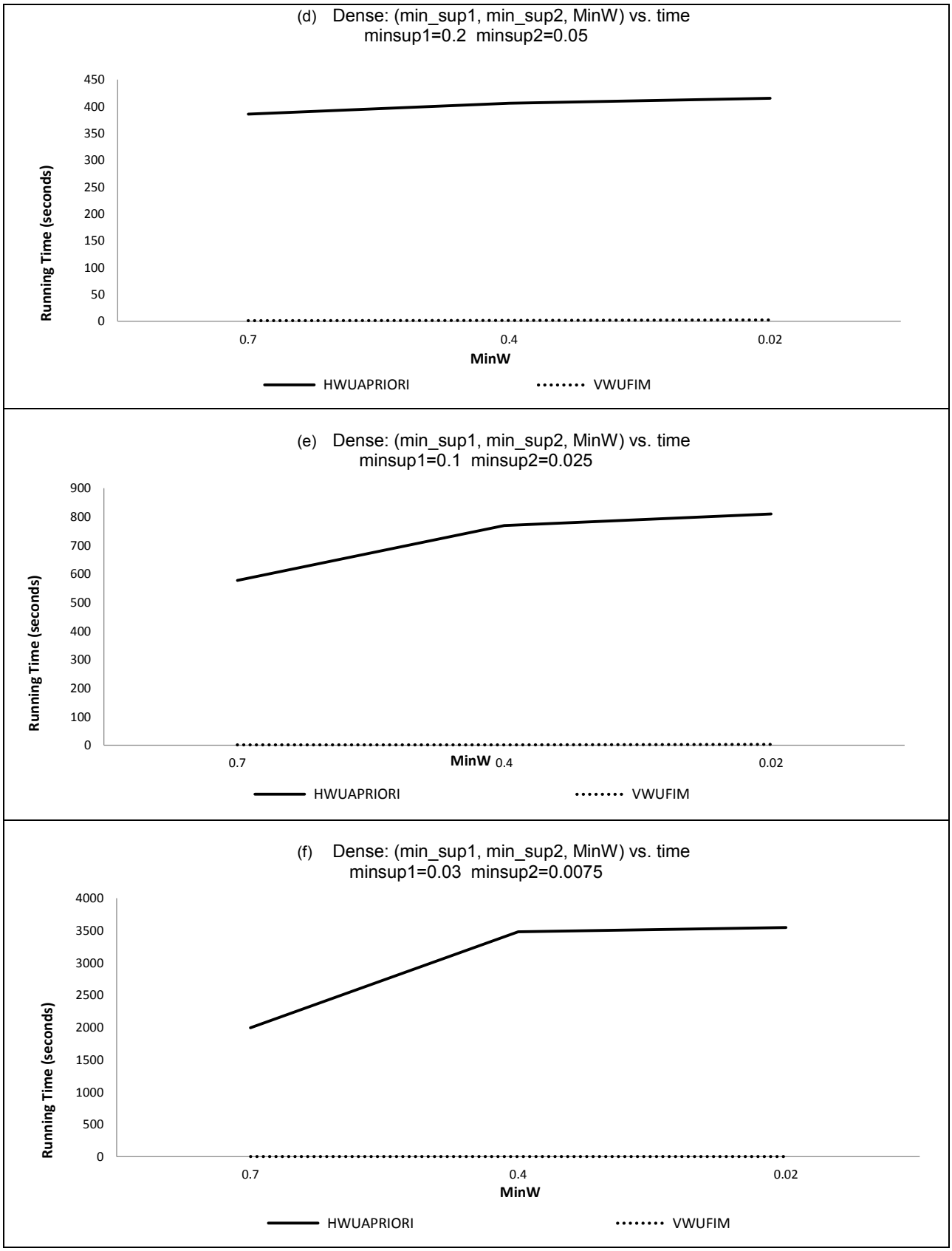


Fig. 3.2. Comparisons between $HWUAPRIORI$ and $VWUFIM$ algorithms

Chapter 4

Disjunctive Rules Mining From Uncertain Data

4.1 introduction

Association rules mining is a common data mining problem that is well researched since its introduction by Agarwal et al. [1] which explores the relationship between items based on their occurrences. A typical algorithm for association rules mining consists of two phases. In the first phase all the frequent itemsets are identified. In the second phase, the frequent itemsets are used to generate association rules. The second phase is relatively simpler compared to the first phase and hence researchers mostly tackle the problem of identifying frequent itemsets.

Consider an example where $I = \{milk, bread, diaper, oil, butter\}$. An association rule of interest could be $\{diaper\} \Rightarrow \{bread, milk\}$. This rule means that if a person buys $diaper$ then (s)he is also likely to buy $bread\ and\ milk$. In this rule the consequent is said to occur when both $bread\ and\ milk$ are present. Thus this rule can be thought of as a conjunctive rule. There are many applications where rules with the antecedent as well as the consequent consisting of disjunctions of itemsets might be relevant. For example the rule $\{diaper\} \Rightarrow \{bread\} or\ \{milk\}$ could be equally important. The implication of this rule is that

if a person buys diaper, then (s)he is also likely to buy either $bread\ or\ milk$. As an example, when one buys a book, say A, from a company such as Amazon, they will inform the customer: "$Persons\ who\ bought\ A\ also\ bought\ B\ or\ C\ or\ \ldots$".

The problem of generating disjunctive rules has been studied well (see e.g., [56, 57]). In [56], a generalized disjunctive rules mining algorithm, called thrifty-traverse, has been proposed. This algorithm generates rules like "People who buy jackets also buy either bow ties or neckties and tiepins". The thrifty-traverse algorithm starts with one-disjunctive rules, and continues growing the rules set until the minimum confidence is satisfied or the specified length of rules is reached. One of the challenges in generating disjunctive rules lies in the need to explore a large collection of possible antecedents and consequents. Existing algorithms have large run times. In [57], an algorithm called DAR has been presented, which aims to filter the not interesting rules and thus avoid generating redundant rules.

All the existing works on disjunctive rules thus far have been carried out to find disjunctive rules on databases without uncertainty. No work has been done on identifying disjunctive rules from uncertain data. By uncertain data we mean a set T of transactions, in which each transaction t is defined as a probability vector $[p_1, p_2, \cdots, p_m]$. Here m is the number of possible items and p_i is the probability that the ith possible item is in t, $\text{for } 1 \leq i \leq m$. The problem of disjunctive rules mining from uncertain data has numerous applications especially in biology, medicine, and bioinformatics. Thus this work fills a gap in the field of rules mining. We present a novel approach that generates disjunctive association rules from uncertain data. We generate rules of length at most k (where k is chosen by the user). Our algorithm can be used to mine disjunctive rules from certain data.

In this case, our algorithm is much simpler than existing algorithms. The run time of our algorithm is comparable to those of the existing algorithms in the worst case while promising to be better in practice. Our algorithm is called $DRMUD$ (Disjunctive Rules Miner from Uncertain Data). The algorithm starts with mining all frequent pairs that satisfy an expected minimum support. Then, it generates disjunctive rules by mining all frequent subsets that satisfy another expected minimum support.

4.1.1 Motivating Applications

There are numerous important applications wherein we have to generate disjunctive rules from uncertain data. We list two of them: 1) In the study of side effects for drugs, the data will consist of transactions where each transaction corresponds to a subject and the transaction itself will consist of a sequence of triplets of the form (d, se, p), where d is a specific drug, se is a specific side effect, and p is the probability that the subject under concern gets the side effect se upon taking drug d. Notice that this is uncertain data. We may not be able to say for sure if a subject will always get the side effect se when taking d. In this data set, disjunctive rules will make more sense since for a given drug d, there could be a set of possible side effects; 2) Consider market data where we have a transaction for each customer (of a specific store). The transaction under concern is a probability vector with an entry for each possible item. The probability refers to the probability that the customer will buy a specific item. This probability can be estimated from the past transactions of the customer. In this uncertain dataset, we may want to generate disjunctive rules.

4.2 Disjunctive Rules Miner from Uncertain Databases (DRMUD)

In this section we present an elegant algorithm for mining disjunctive rules from uncertain databases [58]. This algorithm can be specialized to generate disjunctive rules from data without uncertainties as well. Before presenting details of our algorithm we define precisely what a disjunctive rule is.

Any rule of the form $X \Rightarrow Y_1\ or\ Y_2\ or\ \cdots or\ Y_{k-1}$, where $X, Y_1, Y_2, \cdots, \text{and}\ Y_{k-1}$ are pairwise disjoint itemsets (k being any integer equal to 2 or more) is what we refer to as a disjunctive rule. Just for simplicity of exposition, in the rest of this paper we focus on disjunctive rules where each of these k itemsets consists of a single item. We refer to any such rule as a k-disjunctive rule. Our algorithm is generic and can be readily extended to the general case.

Consider a k-disjunctive rule: $a \Rightarrow b_1\ or\ b_2\ or\ \cdots or\ b_{k-1}$. This rule suggests that if the item a occurs in a transaction then one or more of the items $b_1, b_2, \cdots, \text{and}\ b_{k-1}$ are also likely to occur (with enough support and confidence). Clearly, if the rule $a \Rightarrow b$ has enough support, then the rule $a \Rightarrow b\ or\ c$ also will have enough support even if the rule $a \Rightarrow c$ has zero support. If the rule $a \Rightarrow c$ does not have at least some minimum support, then the rule $a \Rightarrow b\ or\ c$ may not be interesting even if the rule has sufficient support. Keeping this mind, we require that, for the rule $a \Rightarrow b_1\ or\ b_2\ or\ \cdots or\ b_{k-1}$ to be interesting, each of the rules $a \Rightarrow b_j, \text{for}\ 1 \leq j \leq (k-1)$, have some minimum support. We introduce two support parameters $minsup1$ and $minsup2$. The rule $a \Rightarrow b_1\ or\ b_2\ or\ \cdots or\ b_{k-1}$ must have a minimum support of $minsup2$ and each rule $a \Rightarrow b_j$ must have a minimum support of $1\ (for\ 1 \leq j \leq (k-1))$. There are two phases in our algorithm. In the first phase we

identify pairs of items that have enough expected support. In the second phase we utilize these pairs to generate k-disjunctive rules.

A. The first phase: Let $min1_{esup}$ be the minimum expected support that is enforced between any a pair of associated items and $min2_{esup}$ be the minimum expected support that is required between an item a, and the set of items $\{b_1, b_2, \cdots, b_{k-1}\}$, for the rule $a \Rightarrow b_1\ or\ b_2\ or\ \cdots or\ b_{k-1}$ to be interesting. Then,

$$min1_{esup} = N \times minsup1$$

$$min2_{esup} = N \times minsup2$$

We scan through the database to generate all possible pairs of items, with an expected support of $\geq min1_{esup}$. Specifically, for each pair of items (a, b), we calculate its expected support as:

$$esup(\{a, b\}) = \sum_{i=1}^{N} p_i(a) \times p_i(b)$$

Where $p_i(x)$ is the probability that the transaction ti has item x (for any item x). A pair (a, b) is frequent if and only if: $esup(a, b) \geq min1_{esup}$. Let F_2 stand for the set of all frequent pairs.

B. The second phase: We utilize F_2 to generate all the k-disjunctive rules as follows. Let a be any item. Let $\{b_1, b_2, \cdots, b_{k-1}\}$ be any $(k-1)$ −itemset (from $I - \{a\}$) such that each pair (a, b_j) is frequent $(for\ 1 \leq j \leq (k-1))$ and also $\sum_{j=1}^{k-1} esup(a, b_j) \geq min2_{esup}$. In this case, we output $a \Rightarrow b_1\ or\ b_2\ or\ \cdots or\ b_{k-1}$ as a k-disjunctive rule.

Example 4.1: Consider the uncertain transaction database UDB shown in table 4.1. Let $minsup1 = 0.01$, and $minsup2 = 0.1$. Let number of transactions $N = 2$, and $k = 4$.

$min1_{esup} = N \times minsup1 = 0.01 \times 2 = 0.02,\ min2_{esup} = N \times minsup2 = 0.1 \times 2 = 0.2$

Table 4.1 uncertain transaction database

TID	Transactions
t_1	a (0.1) b (0.2) c (0.5) d (0.9)
t_2	a (0.8) b (0.4) d (0.3) g (0.1)

A. First Phase: Identification of frequent pairs

$esup(a,b) = \sum_{i=1}^{N} p_i(a) \times p_i(b) = (0.1 \times 0.2) + (0.8 \times 0.4) = 0.34 > min1_{esup}.$

$esup(a,c) = (0.1 \times 0.5) = 0.05 > min1_{esup}.$

$esup(a,d) = (0.1 \times 0.9) + (0.8 \times 0.3) = 0.33 > min1_{esup}.$ In a similar manner, we realize that the following pairs are also frequent: $(a,g), (b,c), (b,d), (b,g), (c,d)$, and (d,g).

B. Second Phase: Rules Generation

Consider the generation of rules in which a is the antecedent. We know that there are 3 items in the consequent. Thus there are 4 possibilities for the consequent, namely, $\{b,c,d\}, \{b,c,g\}, \{b,d,g\},$ and $\{c,d,g\}$. For each of these possibilities we check if there is enough support. Calculate the expected support for each of the above 4 possibilities.

$esup\ (a,\{b,c,d\}) = esup(a,b) + esup(a,c) + esup(a,d) = 0.72$.

Since $esup\ (a,\{b,c,d\}) > min2_{esup}$, we output the rule: $a \Rightarrow b\ or\ c\ or\ d$.

$esup\ (a,\{b,c,g\}) = esup(a,b) + esup(a,c) + esup(a,g) = 0.47$.

Since $esup\ (a,\{b,c,g\}) > min2_{esup}$, we output the rule: $a \Rightarrow b\ or\ c\ or\ g$.

$esup\ (a,\{b,d,g\}) = esup(a,b) + esup(a,d) + esup(a,g) = 0.75$.

Since $esup\ (a,\{b,d,g\}) > min2_{esup}$, we output the rule: $a \Rightarrow b\ or\ d\ or\ g$.

$esup\ (a,\{c,d,g\}) = esup(a,c) + esup(a,d) + esup(a,g) = 0.46$.

Since $esup\ (a,\{c,d,g\}) > min2_{esup}$, we output the rule: $a \Rightarrow c\ or\ d\ or\ g$.

In a similar manner we can generate all the disjunctive rules for which the antecedent is $b, c, d, or\ g$. We provide below a pseudo code for our algorithm.

4.2.1 DRMUD Algorithm

Our proposed algorithm as follows;

Input:

- *A transaction set T with $|T| = N$,*
- *A set I of items in the transactions,*
- *$minsup1, minsup2$,*
- *Size k of Rules,*

Output:

All the k-disjunctive rules of the form: $(a \Rightarrow b_1\ or\ b_2\ or\ \cdots or\ b_{k-1})$.

Algorithm:

Step 1: Pre-Processing

$min1_{esup} = N \times minsup1$

$min2_{esup} = N \times minsup2$

Step 2: ***/*Find the set F_2 of frequent pairs of items with a support of ≥ min1esup*/***

<u>for</u> every i ∈ I do

<u>for</u> every j ∈ I and (j≠ i) do

if i and j are in at least one transaction together, calculate $esup(i,j)$;

if $(esup(i,j) \geq min1_{esup})$*, store the frequent pair (i,j) in* F_2 together with *esup(i, j)*

end for

end for

Step 3: Generate the k-disjunctive rules

for every a $\in$ *I do*

for every (k-1) subset S of I-{a} do

Let S= $\{b_1, b_2, \cdots, b_{k-1}\}$;

Support=0;

for $1 \leq j \leq k-1$ *do*

$Support \mathrel{+}= esup(a, bj)$;

/ esup values are in* F_2 **/*

endFor

if $Support \geq min2_{esup}$

Output $a \Rightarrow b_1 \; or \; b_2 \; or \; \cdots or \; b_{k-1}$

endIf

endFor

endFor

Note: In Step 3, while considering the set S= $\{b_1, b_2, \cdots, b_{k-1}\}$, if there is any j $(1 \leq j \leq k-1)$ such that $esup(a, bj) < min1_{esup}$, then the set S is not considered.

4.3 Complexity Analysis

The run time of Step 2 is $O(m^2N)$, where m is the number of items and N is the number of transactions. The run time of Step 3 is $O\left(mk\binom{m-1}{k-1}\right) = O(km^k)$. Therefore, the total run time of the algorithm is $O(m^2N + km^k)$. Please note that the existing algorithms [56, 57] for disjunctive rules mining have similar run times in the worst case. Specifically, their run times are also exponential in k. Our algorithm is significantly simpler than those of [56] and [57].

4.4 Experimental Results

In this section, we present some experimental results on $DRMUD$. $DRMUD$ algorithm is designed using the horizontal layout. It is implemented and compiled using Microsoft's Visual Studio C++ 2013, and it was run on an Intel(R) Core(TM) i7 3.40GHz PC with 8GB Main memory, operating on Microsoft Windows 7.

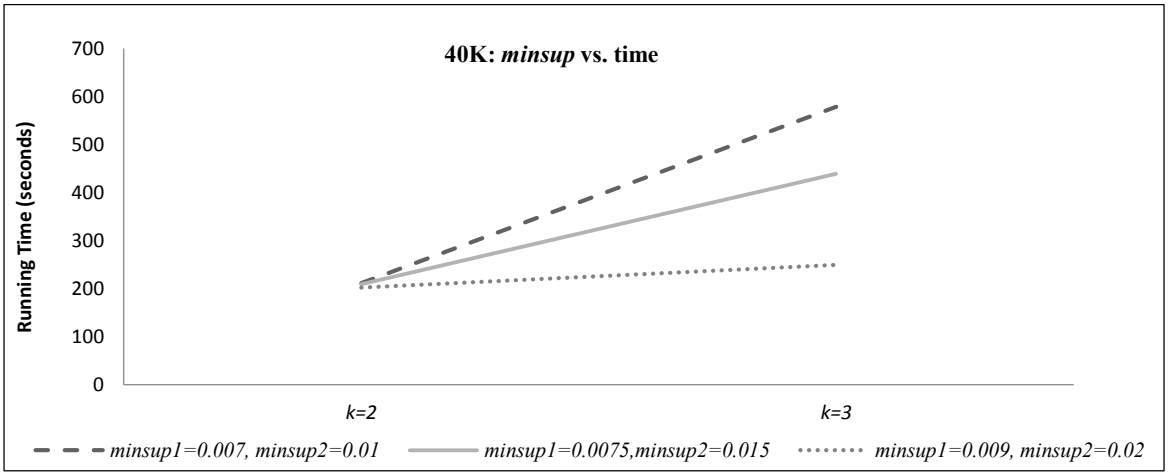
We used four datasets: one sparse dataset and three dense datasets for testing the performance of $DRMUD$ over an uncertain database; $Gazelle$ dataset is a sparse dataset obtained from [44]. It consists of 59,602 transactions with 497 different items. $synthData$ dataset is a synthetic dataset which we generated with 10k transactions associated with 20 different items. The expected support for each item in the transaction was generated as a discrete random variable. The expected occurrence of each item in each transaction was 0.5. The $third\ and\ fourth\ datasets$ are dense datasets obtained from [44]. These two datasets have 20,000, and 40,000 transactions, respectively, associated with 994 different items. For each dataset, we have run our algorithm with different values of k, and

different configuration parameters: $minsup1$, and $minsup2$. We have listed the total time taken to generate the k-disjunctive rules for various user-supplied values of k, $minsup1$, and $minsup2$ (see figure 4.1).

Since there are no existing algorithms for generating disjunctive rules from uncertain data, we could not compare $DRMUD$ with any existing algorithm. $DRMUD$ can easily be modified to mine disjunctive rules from data without uncertainty. In this case also, we could not compare our algorithm with those of [56] and [57] since we could not access the associated programs.

Table 4.2: Result of horizontal $DRMUD$ under Dense dataset, 40K trans. with 994 Items

Thresholds	k=2	k=3
minsup1=0.007, minsup2=0.01	211.52	578.88
minsup1=0.0075, minsup2=0.015	209.23	439.28
minsup1=0.009, minsup2=0.02	202.71	249.77



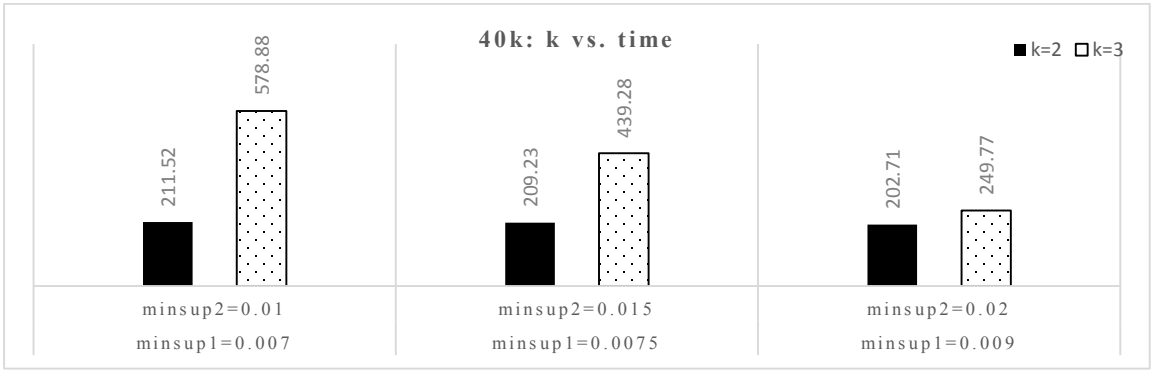
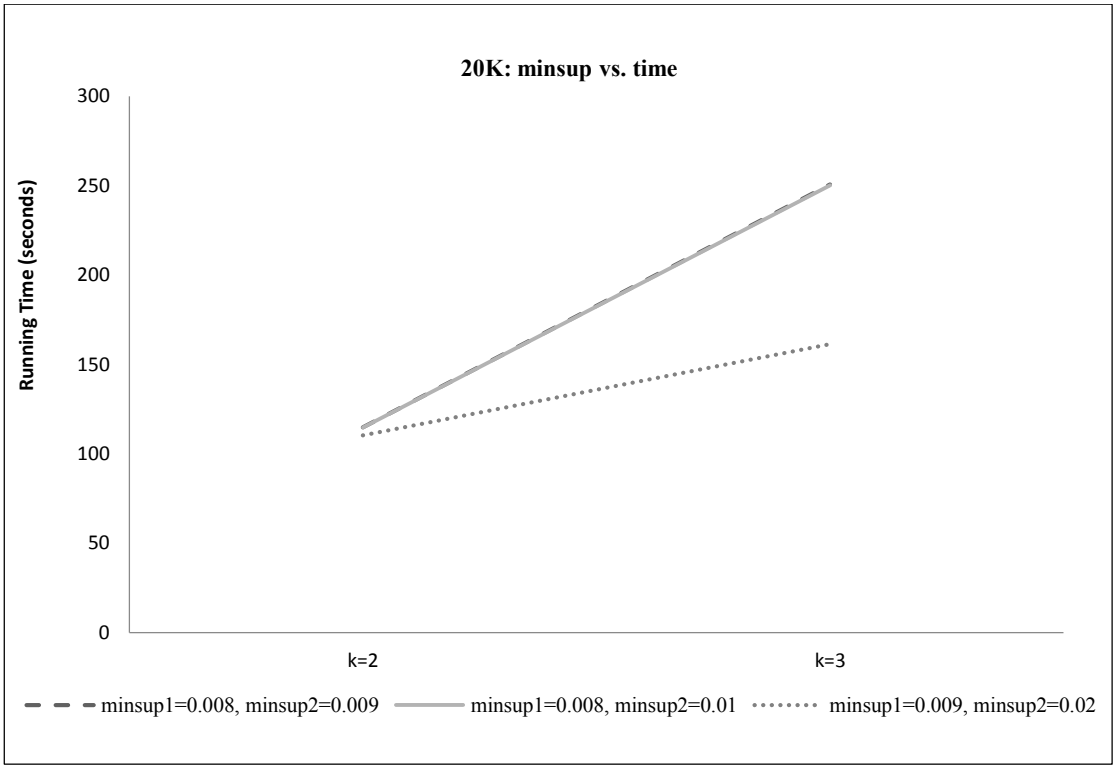


Table 4.3: Result of horizontal $DRMUD$ under Dense dataset 20K trans. with 994 items

Thresholds	*k=2*	*k=3*
*minsup*1=0.008, *minsup*2=0.009	114.86	250.77
*minsup*1=0.008, *minsup*2=0.01	114.82	250.33
*minsup*1=0.009, *minsup*2=0.02	110.32	161.43



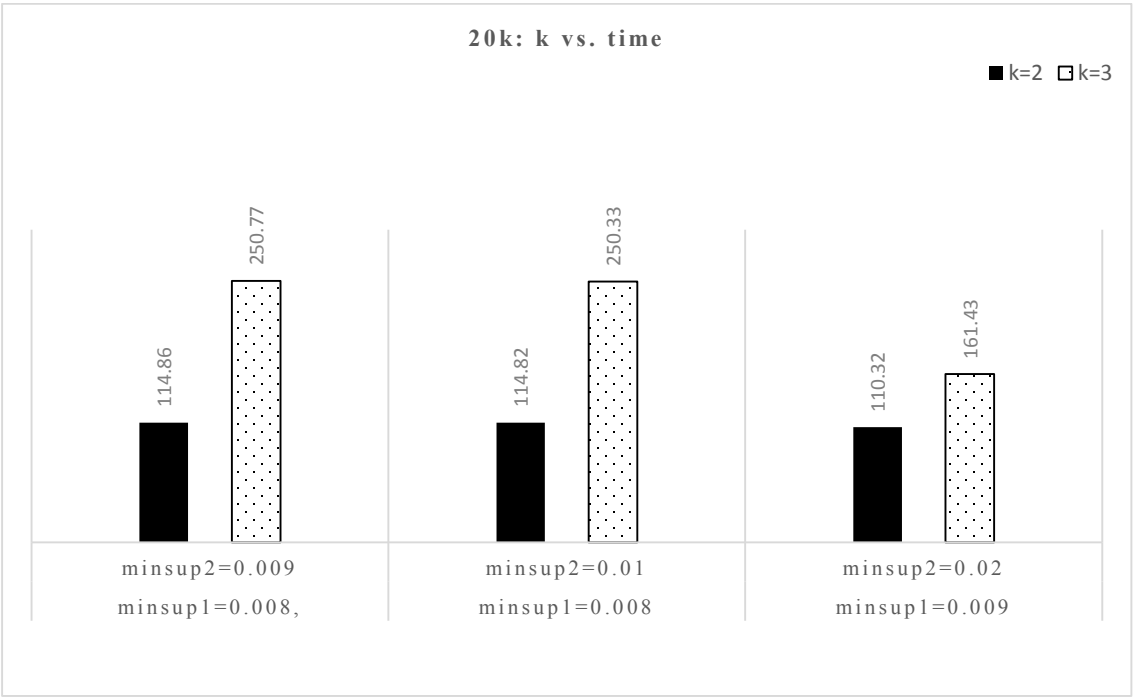
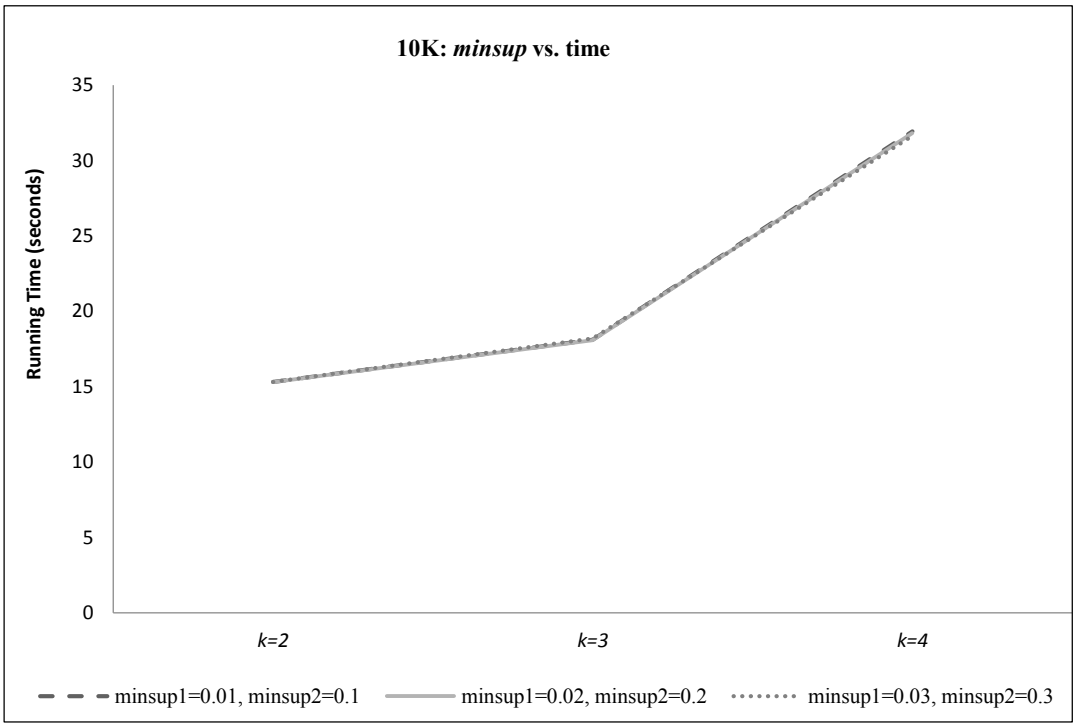


Table 4.4: Result of horizontal $DRMUD$ under synthData dataset, 10K Trans. with 20 Items

Thresholds	k=2	k=3	k=4
minsup1=0.01, minsup2=0.1	15.32	18.11	31.95
minsup1=0.02, minsup2=0.2	15.32	18.09	31.83
minsup1=0.03, minsup2=0.3	15.31	18.21	31.66



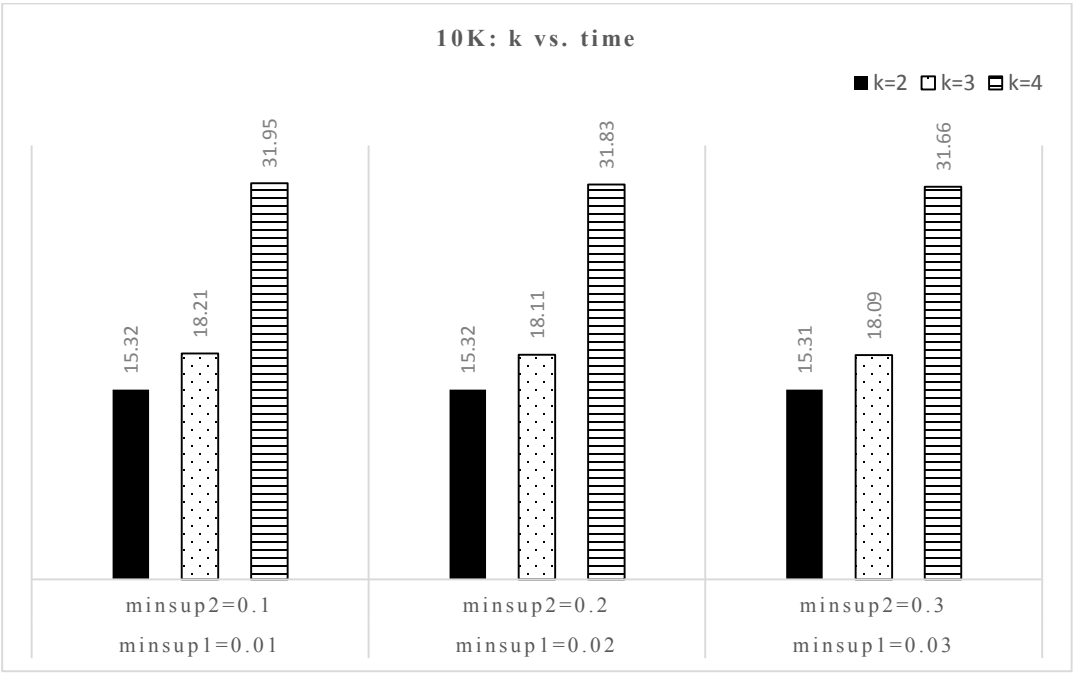
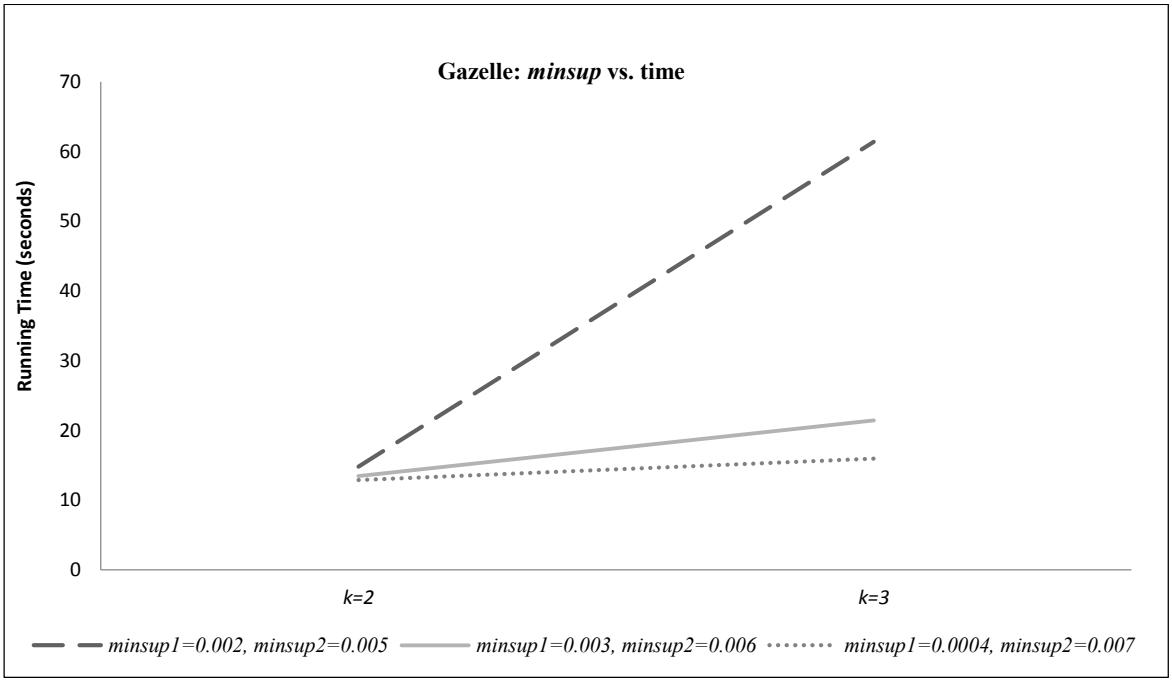


Table 4.5: Result of horizontal $DRMUD$ under Gazelle sparse dataset, 59602 Trans. with 994 Items

Thresholds	k=2	k=3
minsup1=0.002, minsup2=0.005	14.83	61.43
minsup1=0.003, minsup2=0.006	13.47	21.43
minsup1=0.0004, minsup2=0.007	12.88	15.98



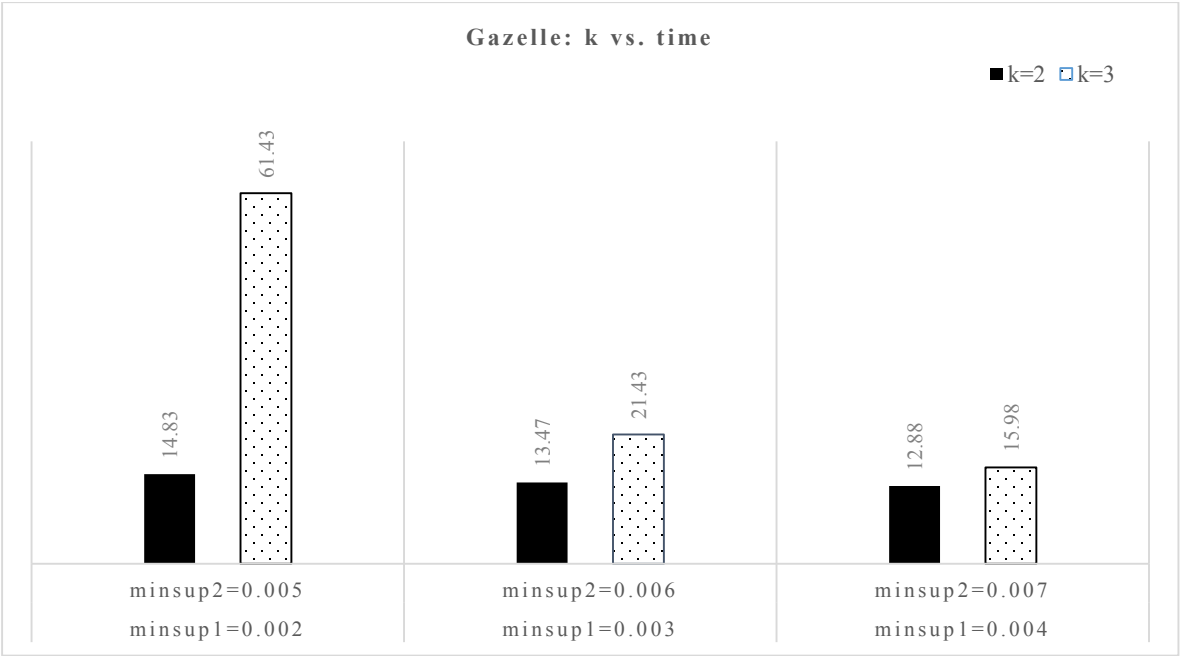


Fig. 4.1. Performance of Horizontal DRMUM algorithm

4.4.1 Horizontal vs. Vertical DRMUD algorithm

In our previous work [58] we showed that the vertical layout based approaches outperform the horizontal layout based approaches in mining weighted frequent patterns for the conjunction case. The vertical format improves the run times for $DRMUD$ as follows:

- Let n be the number of transactions
- q = average number of items in a transaction
- m = total number of items
- ρ = average size of each item list
- I^k= a generic *k*-itemset
- $T(I^k)$ = a list of transactions that I^k appears in.
- F^k = set of all *k*-frequent itemsets
- Let large K be the size of the rule.

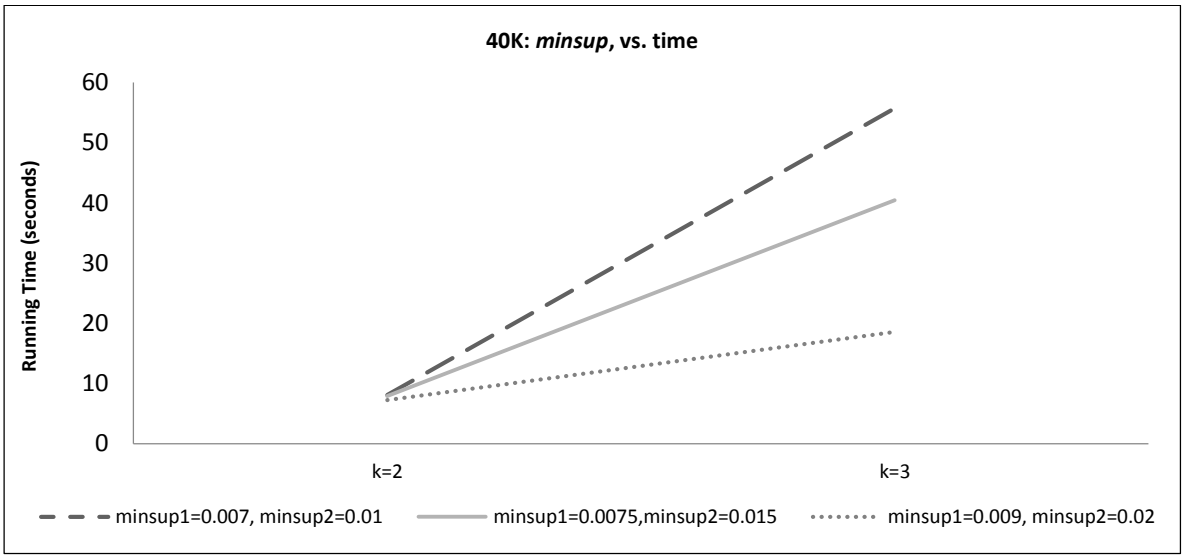
First converting data from horizontal to vertical format takes $O(q * n)$ time. Step 2 takes $O\left(\binom{m}{k} p\right)\ time$, where it takes $O(\binom{m}{2} p)$ time for a 2-frequent itemset, and testing all pairs requires an expected time of $O\left(\binom{m}{k} p\right)$. The run time of Step 3 is $O\left(mk\binom{m-1}{k-1}\right) = O(km^k)$. Therefore, the total run time of the algorithm is $O\left(\binom{m}{k} p + km^k + qn\right)$.

In this section, we compare $DRMUD$ under the two approaches. For the sake of comparison, we used the same setting that we used for horizontal $DRMUD$ algorithm. We also used the same datasets and ran our vertical algorithm with the same values of k, and configuration parameters: $minsup1$, and $minsup2$ that we used in horizontal format. Figure

4.2 shows the performance of our vertical $DRMUD$ under different datasets, dense and sparse, different configuration parameters: $minsup1$, and $minsup2$, and different values of k. Figure 4.3 shows a comparison between the vertical and horizontal versions of $DRMUD$ in terms of run times, where the runtimes differ by the parameters specified by the users.

Table 4.6: Result of vertical $DRMUD$under Dense dataset, 40K trans. with 994 Items

Thresholds	k=2	k=3
minsup1=0.007, minsup2=0.01	8.1	55.73
minsup1=0.0075, minsup2=0.015	7.9	40.46
minsup1=0.009, minsup2=0.02	7.27	18.6



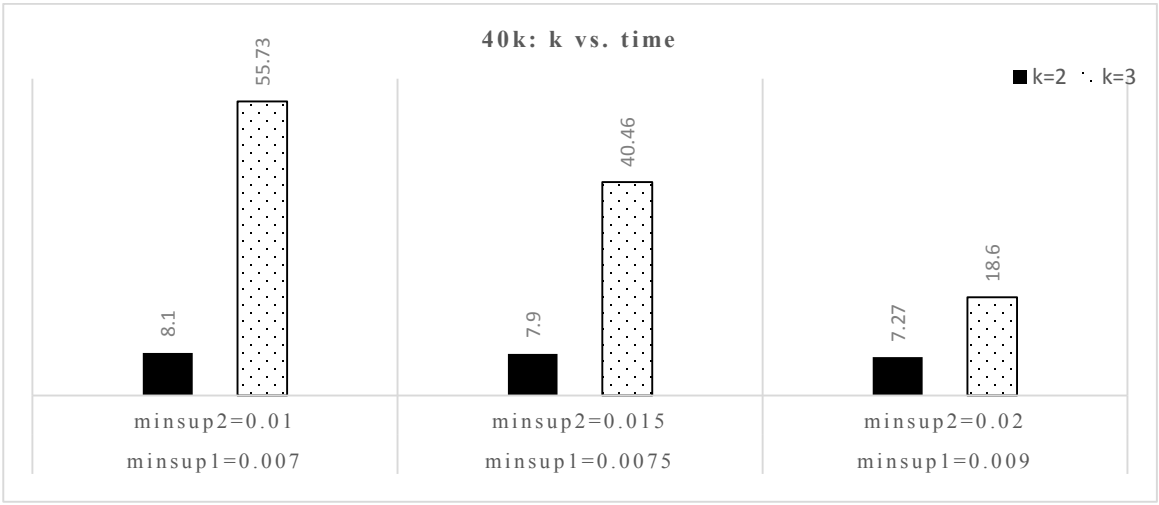
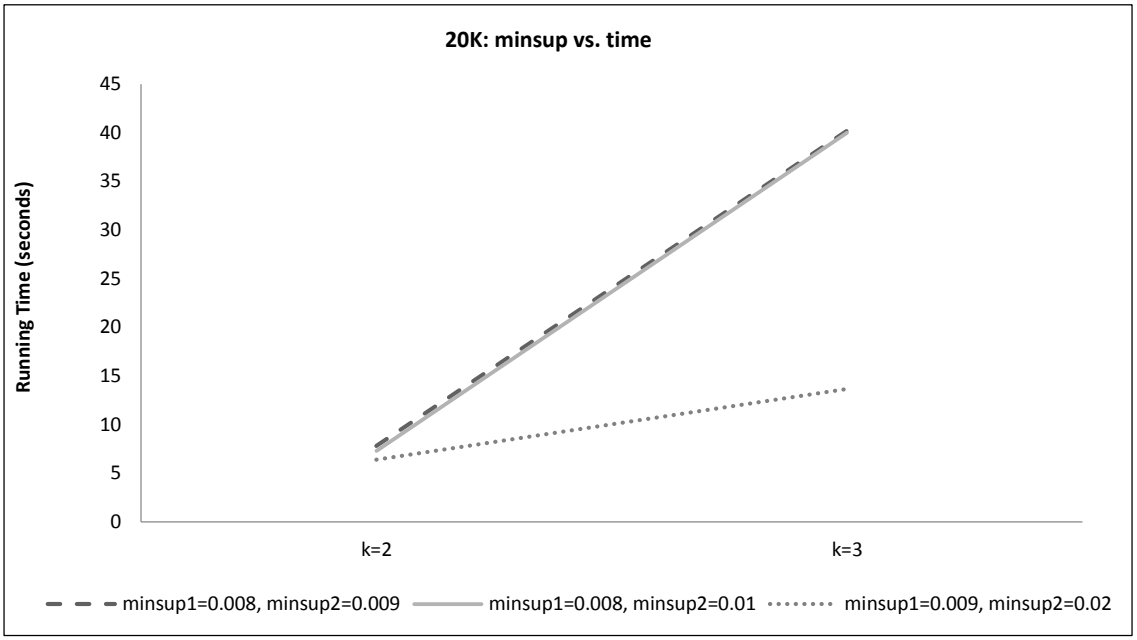


Table 4.7: Result of vertical $DRMUD$under Dense dataset 20K trans. with 994 items

Thresholds	*k=2*	*k=3*
*minsup*1=0.008, *minsup*2=0.009	7.8	40.2
*minsup*1=0.008, *minsup*2=0.01	7.29	40
*minsup*1=0.009, *minsup*2=0.02	6.38	13.67



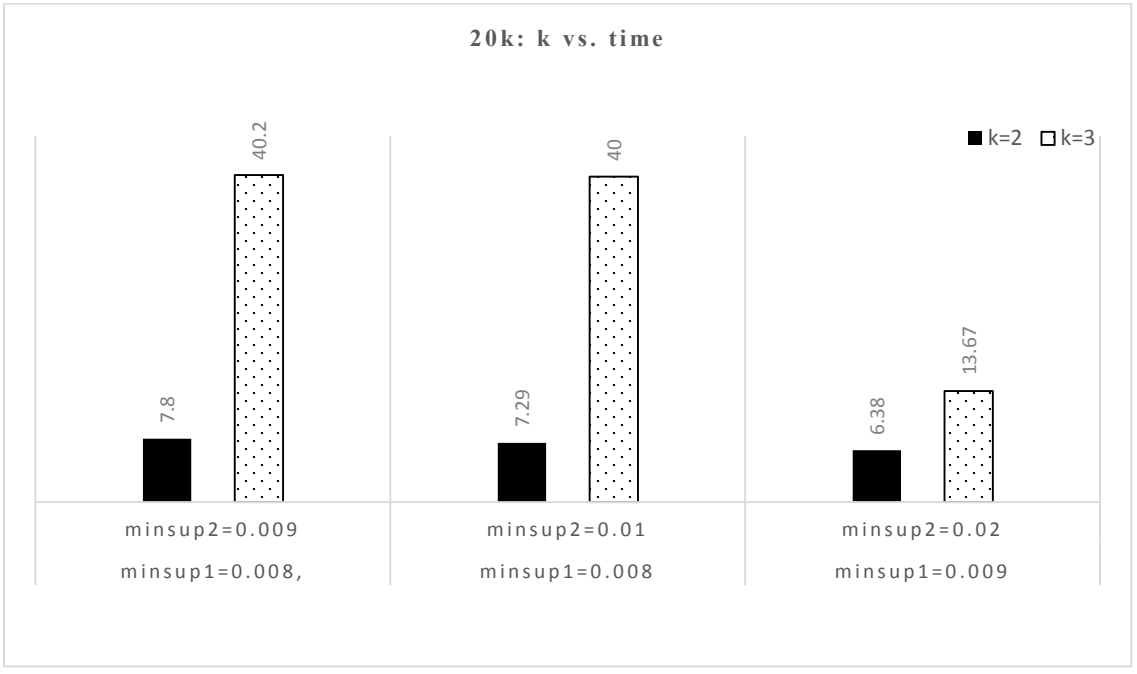
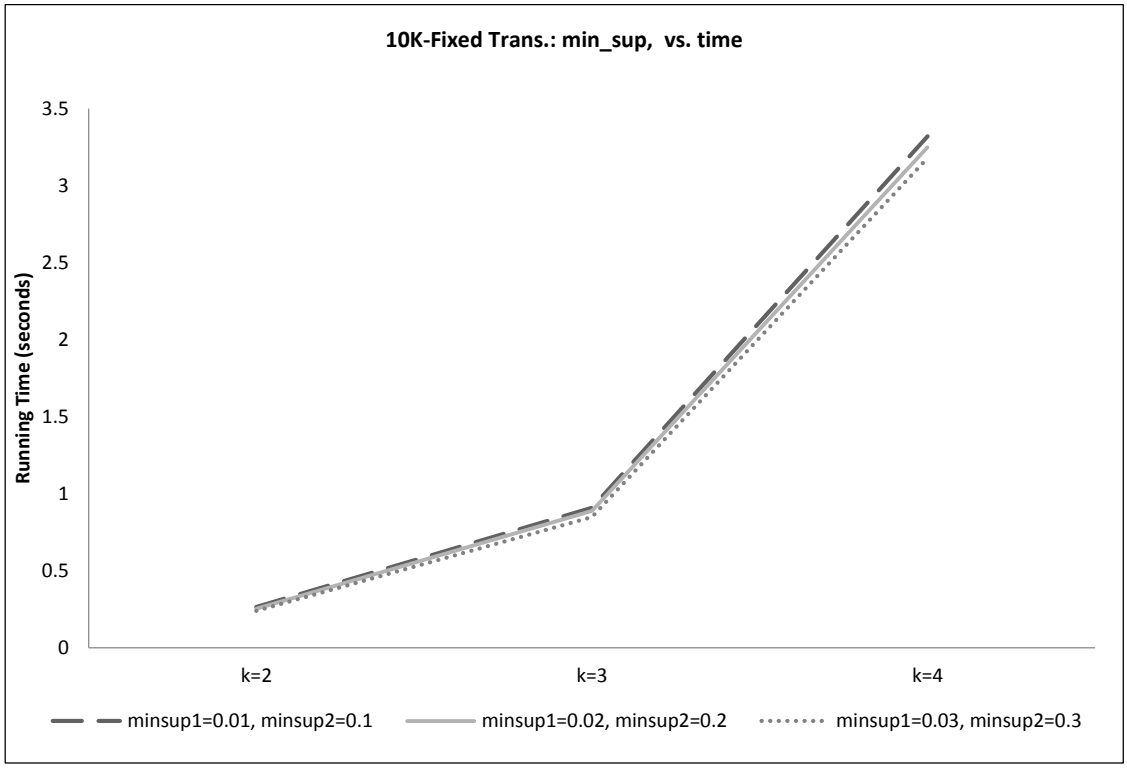


Table 4.8: Result of vertical $DRMUD$under synthData Dense dataset, 10K Trans. with 20 Items

Thresholds	k=2	k=3	k=4
minsup1=0.01, minsup2=0.1	0.265	0.91	3.32
minsup1=0.02, minsup2=0.2	0.253	0.89	3.25
minsup1=0.03, minsup2=0.3	0.241	0.85	3.18



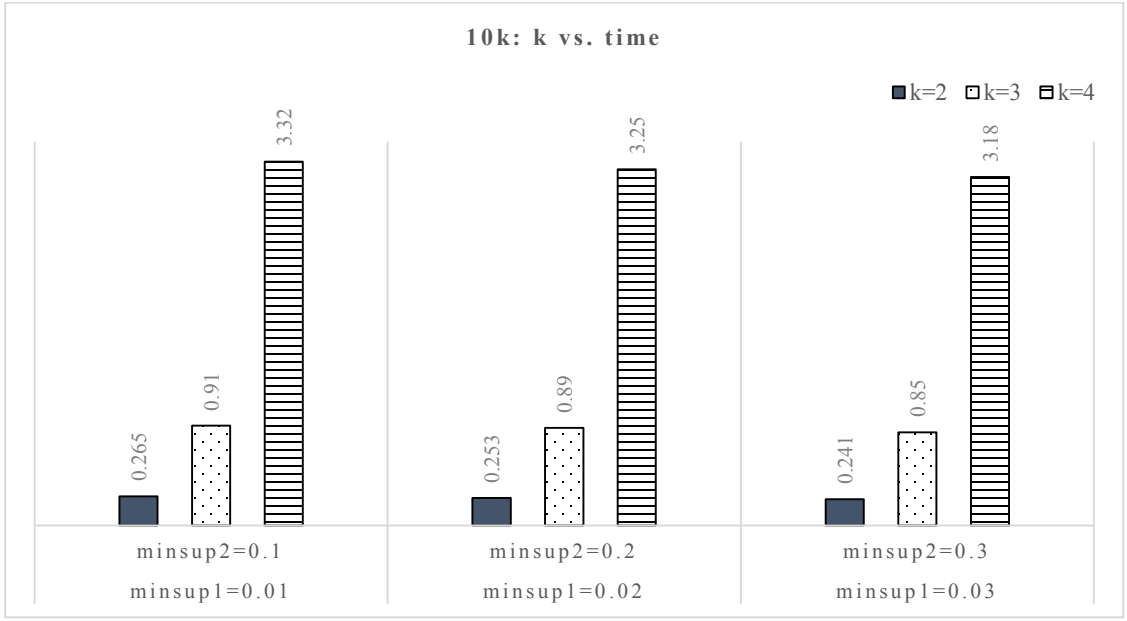
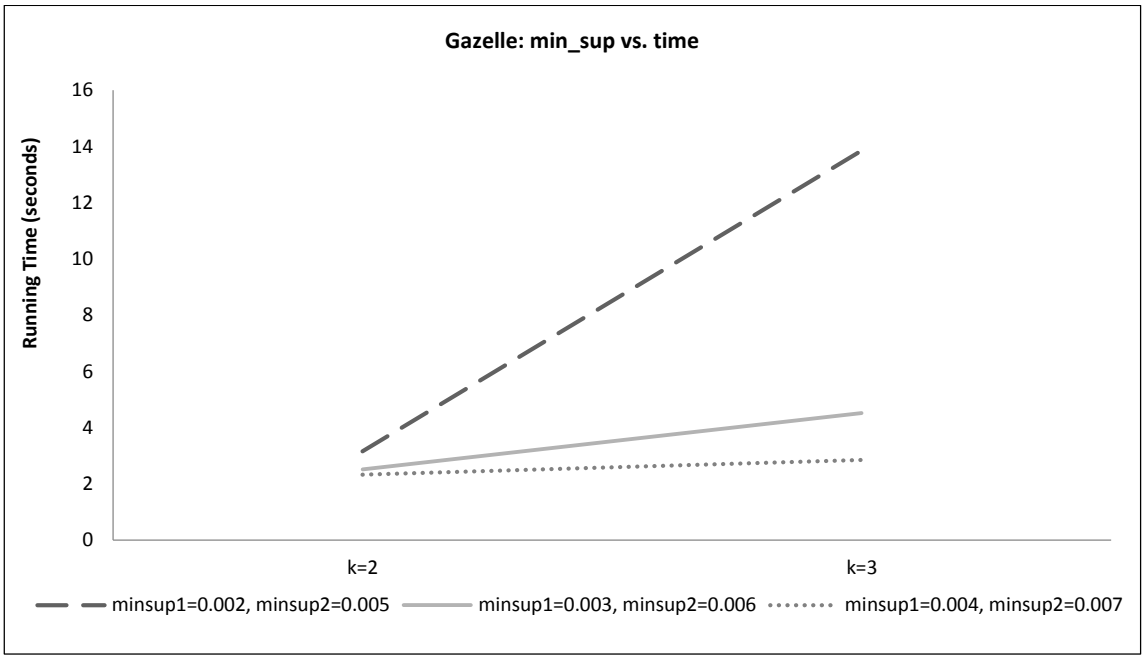


Table 4.9: Result of vertical $DRMUD$under Gazelle sparse dataset, 59602 Trans. with 994 Items

Thresholds	k=2	k=3
minsup1=0.002, minsup2=0.005	3.16	13.85
minsup1=0.003, minsup2=0.006	2.52	4.52
minsup1=0.0004, minsup2=0.007	2.33	2.85



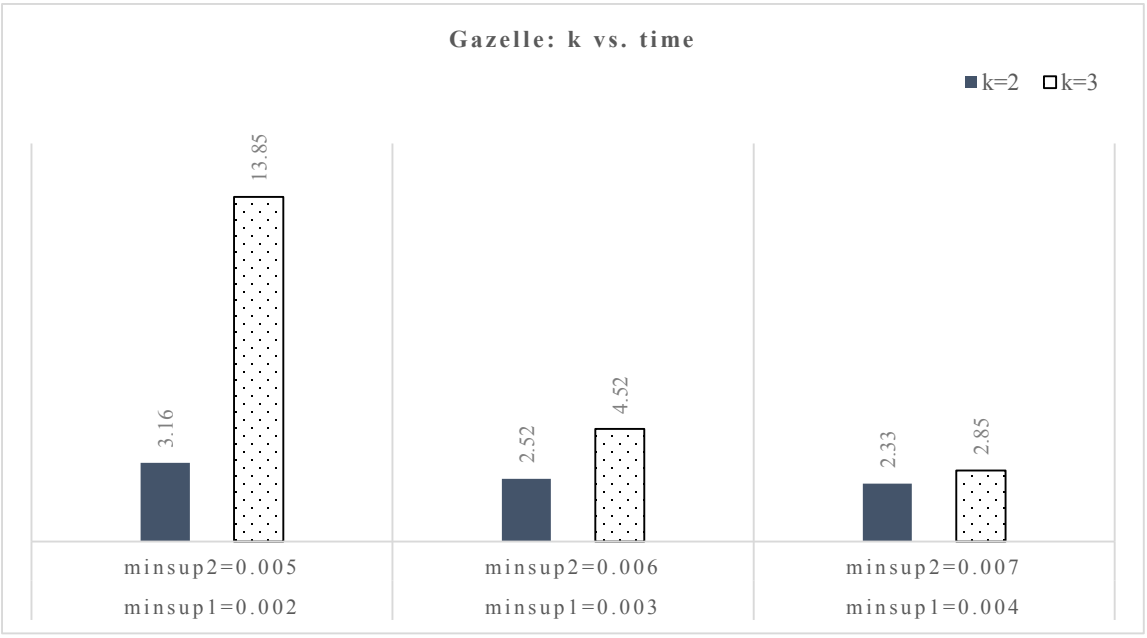
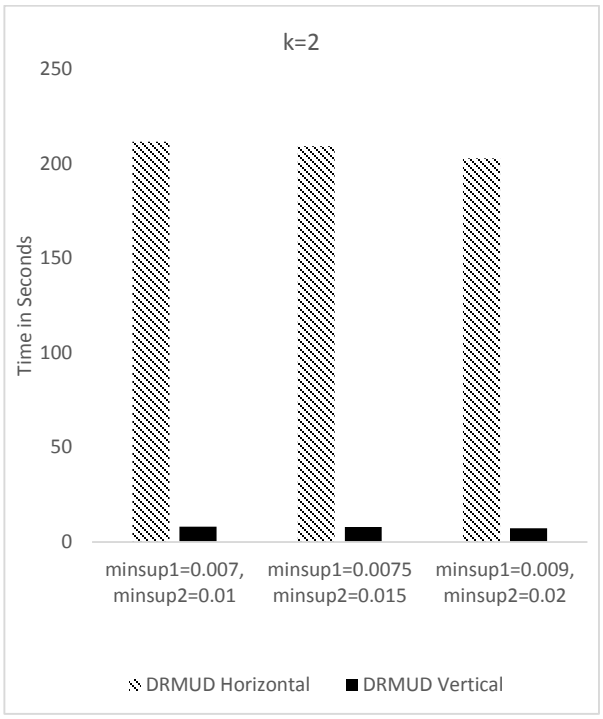
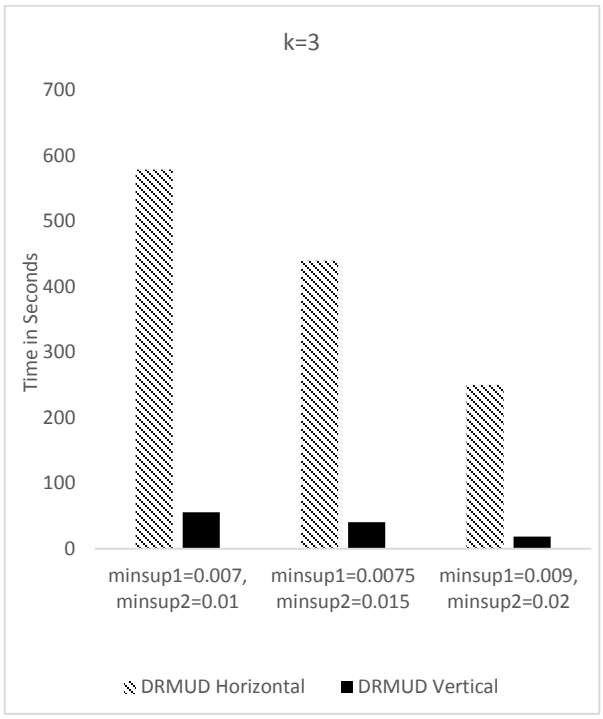


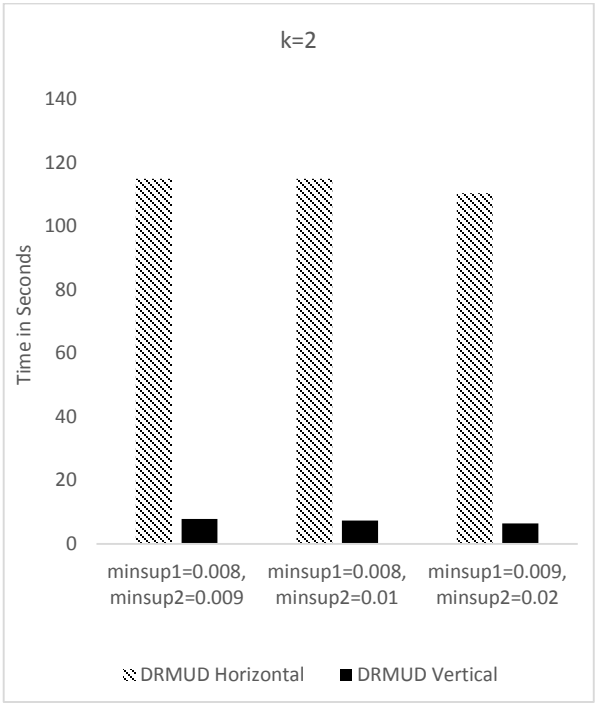
Fig. 4.2. Performance of Vertical DRMUD algorithm

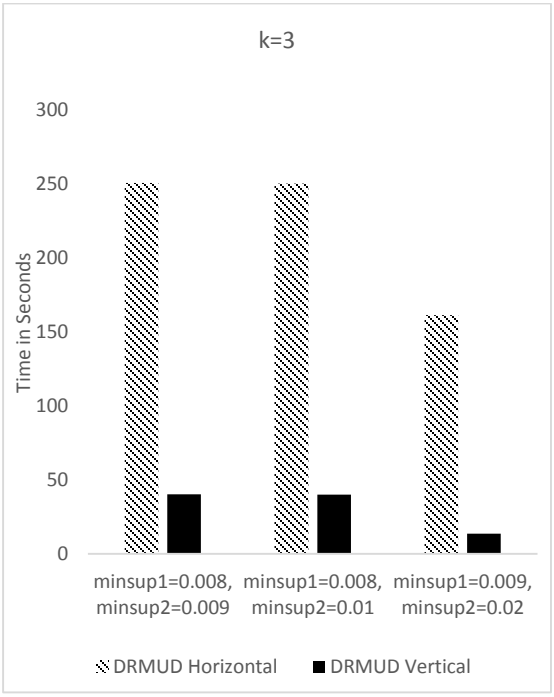
a) Comparisons of Horizontal and Vertical $DRMUD$ algorithms under Dense dataset, 40K transactions with 994 Items



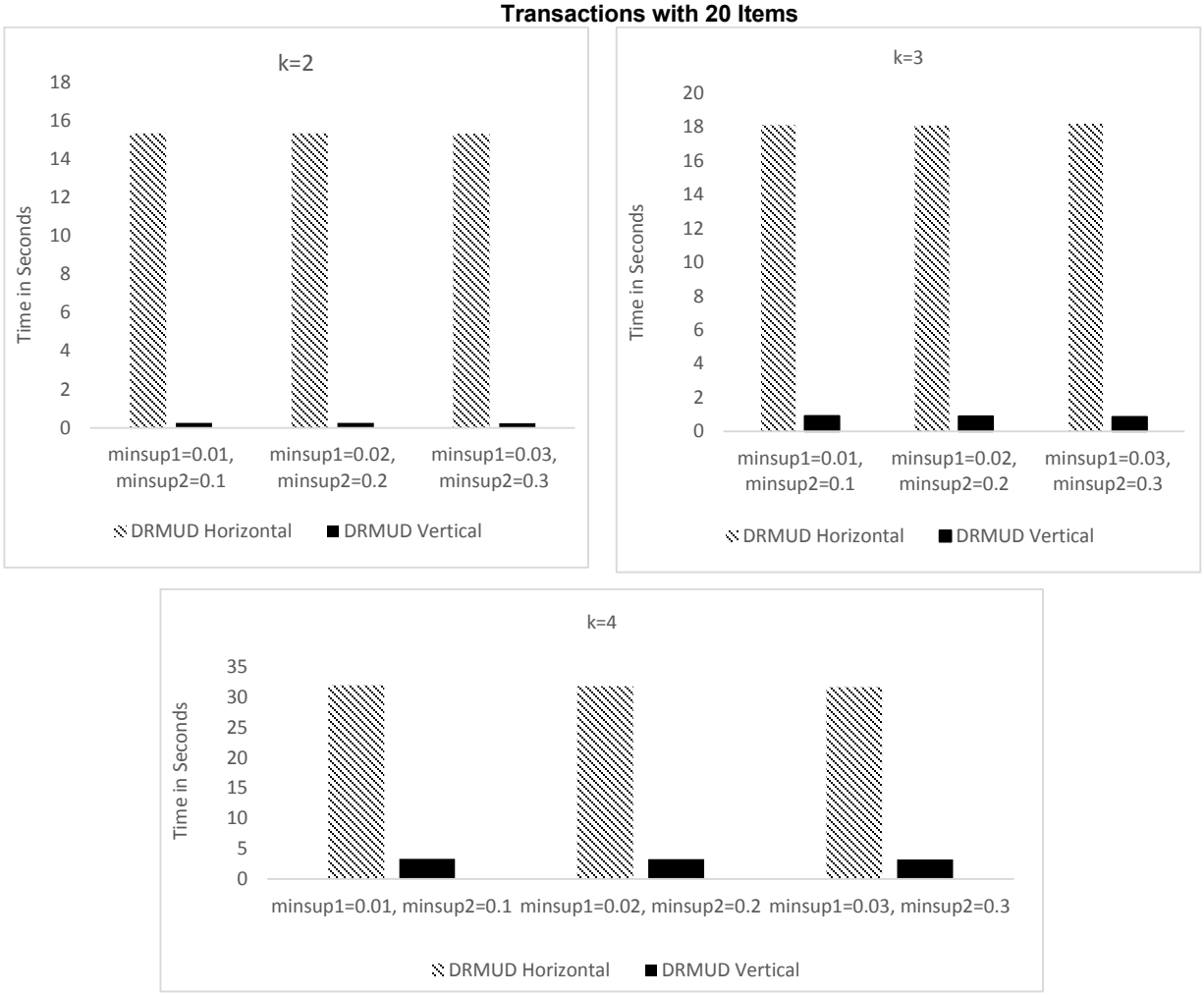


b) Comparisons of Horizontal and Vertical $DRMUD$ algorithms under Dense dataset, 20K transactions with 994 Items





c) Comparisons of Horizontal and Vertical $DRMUD$ algorithms under synthData Dense dataset, 10K Transactions with 20 Items



d) Comparisons of Horizontal and Vertical $DRMUD$ algorithms under Gazelle sparse dataset, 59602 Transactions with 994 Items

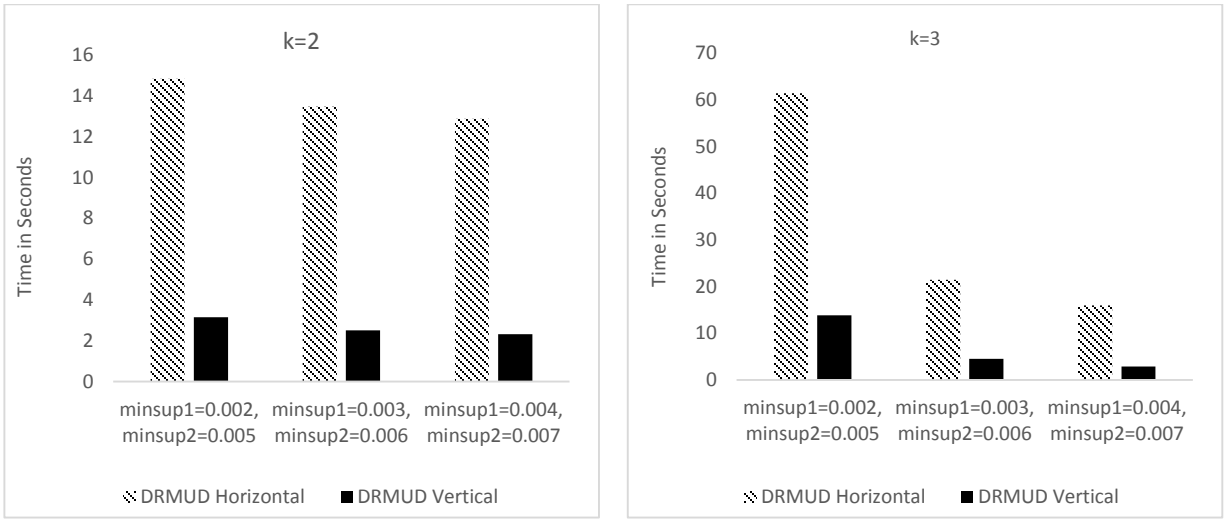


Fig. 4.3. Comparisons between Horizontal and Vertical for DRMUD algorithms

4.5 Conclusions

Traditional conjunctive association rules mining algorithms may not be suitable for all applications. There are many crucial applications for which there is some uncertainty in the data and also disjunctive rules are called for. Algorithms can be found in the literature for mining disjunctive rules from data without uncertainty. Algorithms also exist for mining conjunctive rules from uncertain data. To the best of our knowledge, no algorithms have been proposed in the literature for mining disjunctive rules from uncertain data. In this paper, we fill this gap by proposing a novel approach that can be used to mine disjunctive rules from uncertain transactional databases. Our algorithm, called $DRMUD$, starts by mining all frequent pairs that satisfy an expected minimum support of $min1_{esup}$. Then, it generates disjunctive rules by mining all frequent subsets that satisfy an expected minimum support of $min2_{esup}$. Our experimental results reveal that $DRMUD$ is effective in generating disjunctive rules from both dense and sparse datasets.

Chapter 5

Causal Rules Mining From Uncertain Data

5.1 Introduction

While traditional association rules mining algorithms identify the relationships among variables in general, the aim of causal discovery is to identify profound relationships such as “a change of antecedent is the cause for a change in the consequent”. Association rules mining uses the well-known support–confidence framework which does not necessarily signify causation [59]. For example, the support–confidence framework can show that milk and eggs in the same basket are related but cannot indicate that buying eggs was caused by buying the milk. Causal relationships discovery is widely used in the prediction processes where the prediction of causes have been used to prevent harmful consequences [60]. The importance of discovering causal relationships can be felt in many areas, such as economics, physical, behavioral, medical, biological, and social sciences. Thus, significant advances in exhibiting and finding causal rules have been made in many areas. One of the most powerful tools for causal discovery is the Randomized controlled trial (RCT) [61]. RCT is an experimental approach, and the main challenge in this approach is the difficulty of conducting experiments (i.e., we can't

control everything) because of ethical concerns (e.g., many experiments on humans like smoking or drinking, would be considered unethical) or cost issues (e.g., experiments that consider a large number of variables like studying the star formation and gene knock-down experiments are very expensive) [62]. Causal discovery with observational data is an alternative solution when the experimental approaches are infeasible. Due to a rapid expansion of observational data, researchers have focused on attempts to reduce costs and help in decision making by predicting vital indicators that prevent harmful consequences [62]. In spite of advances made in finding causal rules, existing methods are unable to handle big datasets. Several variations of causal discovery with observational data have been investigated in the literature. Most of the previous works were based on statistics [61-80]. The main challenge in these approaches on observational data is that statistical correlations discovered from observational data may not really form a causal relationship [62]. In the next section, we highlight the most prominent studies in this area.

5.1.1 Causal Discovery

Most of the existing studies [63-74] mainly focus on inferring causal relationships in observational data using a directed acyclic graph (Bayesian networks) or undirected probabilistic graphical models (Markov networks). It is known that Bayesian networks based formulations are NP-hard and therefore algorithms based on these are only applied on low dimensional data sets [74]. Constraint based approaches have been used as optimization methods as they do not search for a complete Bayesian network [75-79]. Unfortunately, they have two problems [62, 82]: they only discover single causes, and they

fail to discover causal relationships on non-fixed structures. Integrating partial association tests along with association rules mining is a solution that has been introduced by [62, 78, and 82]. These solutions link causality with continuity, where the association between two variables is not affected by other variables. For example, it is reasonable to conclude that change of gender is the cause for salary differences, if there is always salary differences between male/female workers whatever the circumstances are (e.g., different ages, domains, and different qualifications). In this case, the association between being female and receiving low salaries holds [62]. The CR-PA algorithm [82] has been proposed to discover causal relationships with both a single cause variable, and multiple cause variables.

The problem of discovering single or combined casual rules where the target is a set of k-disjunctive variables or a set k- conjunctive variables is of interest in many different fields including medicine, epidemiology, and bioinformatics. An example will be that of mining from any drug side effects database, where each record corresponds to a subject and the record itself consists of a sequence of *quadruplets* of the form (d, cer, se, p), where d is a specific drug, cer is a set of circumstances, se is a specific side effect, and p is the probability that the subject under concern gets the side effect se upon taking drug d within certain circumstances.

In this research work, we are interested in discovering disjunctive as well as conjunctive causal rules from uncertain data. We are concerned with single as well as multiple causes. We are interested in reducing the cost of causal discovery by employing the study of frequent itemsets mining to discover conjunctive combined causal rules from uncertain data. We propose novel approaches that adopt some of the strategies from previous

works on association rules mining [78] and partial association rules mining [82] to discover causal relationships in observational data. In specific, we propose two algorithms;

1. *DCCRUD* algorithm for discovering disjunctive causal rules mining from uncertain data. It uses some ideas from our previous work [58]. *DCCRUD* discovers disjunctive combined causal rules from uncertain data.
2. *CCCRUD* algorithm for discovering conjunctive causal rules mining from uncertain data. We are concerned with single as well as multiple causes. It also uses some ideas from our previous work [55]. CCCRUD discovers conjunctive combined causal rules from uncertain data.

Earlier works have found that the vertical layout outperforms the horizontal layout in mining frequent patterns. Thus, our algorithms also use vertical layout to speed up the process of generating the candidate rules. They discover conjunctive as well as disjunctive combined causal rules from uncertain data. Prior algorithms find causation (single or combined) with a single target. In contrast, our target consists of *k-disjunctive* or *k-conjunctive* variables. We evaluate the performance of the proposed algorithms on real datasets. In the next section, we highlight the most prominent studies in this area.

5.2 Definitions

In this section, we present some basic definitions that apply for mining frequent itemsets from an uncertain database. We introduce the statistical definition of correlation between two variables, and present the concepts for inferring causality from partial associations.

5.2.1 Correlations vs. Traditional Support-Confidence Framework

Traditional support-confidence associations use downward closure property to reduce the search space. Downward closure property states that any subset of a frequent itemset is also frequent which implies that any superset of an infrequent itemset is also infrequent [1]. In reality support-confidence framework might be misleading since it ignores the negative correlations [83]. For example, if the confidence of the rule $(A \rightarrow B)$ is 80%, and the support for buying only product B $(i.e., \sup(B))$ is 90%), then buying product A is 10% less likely than buying product B. The well-known *Chi-square* statistic test is widely used for testing the correlation or the significance of the association between variables. Table 5.1 shows $2x2$ contingency table that is used to calculate the Chi-square for two itemsets $\boldsymbol{x}$ and $\boldsymbol{y}$, where n_{11} is the number of transactions in which $\boldsymbol{X}$ and $\boldsymbol{Y}$ are present together, n_{12} is the number of transactions in which only $\boldsymbol{X}$ is present and $\overline{\boldsymbol{Y}}$ is absent, n_{21} is the number of transactions in which only $\boldsymbol{Y}$ is present and $\overline{\boldsymbol{X}}$ is absent, and n_{22} is the number of transactions in which both $\overline{\boldsymbol{X}}$ and $\overline{\boldsymbol{Y}}$ are absent**.** When any of the cells $\{n_{11}, n_{12}, n_{21}, n_{22}\}$ in the contingency table is dependent (if its value differs sufficiently from its expected value), $\boldsymbol{x}$ and $\boldsymbol{y}$ are said to be correlated. We can calculate the expected values using the following equations.

$$E(n_{11}) = c_1 \times \frac{r_1}{n},\ E(n_{12}) = c_2 \times \frac{r_1}{n},\ E(n_{21}) = c_1 \times \frac{r_2}{n},\ E(n_{22}) = c_2 \times \frac{r_2}{n}.$$

Once we calculate the expected value for each cell, we can easily calculate the chi-squared statistic as $\mathcal{X}^2 = \sum_{i=1}^{2}\sum_{j=1}^{2}\frac{(n_{ij} - E(n_{ij}))^2}{E(n_{ij})}$. Two items are said to be correlated or dependent when its Chi-square value $\mathcal{X}^2$ is higher than a significance threshold $\mathcal{X}_p^2$. When

p-value=*0.05*, we can have 95% confidence to reject the hypothesis that the two items are independent. In other words, two items are dependent (correlated/associated) with each other. A ***p-value*** of *0.05* is considered as a borderline between significant and not-significant results.

Table 5.1: $\mathbf{2} \times \mathbf{2}$ Contingency Table

LHS	**RHS**		**Total**
	Y	$\overline{\boldsymbol{Y}}$	
X	n_{11}	n_{12}	$r_1 = n_{11} + n_{12}$
$\overline{\boldsymbol{X}}$	n_{21}	n_{22}	$r_2 = n_{21} + n_{22}$
Total	$c_1 = n_{11} + n_{21}$	$c_2 = n_{12} + n_{22}$	$n = n_{11} + n_{12} + n_{21} + n_{22}$

Unlike the traditional *support-confidence* associations that take advantage of downward closure property, correlation using *Chi-square* statistic test is upward closed in the itemset lattice. Upward closure property is a property of dependence (correlation) that states that if an itemset *A* is dependent, then its superset will be also dependent. Thus, the search space will be reduced since adding items to a correlated itemset *A* will not cancel the correlation [83]. The *Chi-square* statistic test is easy to calculate. However, it has two major limitations: first, the expected values in all the cells in the contingency table must have a value greater than one; second, the expected values in at least 80% of the cells in the contingency table must be greater than 5. Brin, et al. [83] solved this problem by using the *support-confidence framework* as an additional pruning condition for finding correlation rules. They also extended the definition of support as follows: first mine the data in a level-wise manner beginning with the deletion of all the items less than a minimum support thresholds ***s***. Generate the candidates of 2-size itemsets, and create the contingency table for each candidate. If less than 25% of the cells have count ***s***, then test another candidate. Otherwise, if the $\mathcal{X}^2$ value is at least $\mathcal{X}_p^2$ then it is dependent, and delete all supersets of this set. If a set is not dependent, then use it as a candidate for dependency. Zhou, et al.

also used the *support-confidence framework* as an additional pruning condition for finding correlation rules [62, 78]. They also, used combined contingency tables for more than two variables instead of using multi-way contingency tables (see table 5.2).

Table 5.2: $\mathbf{2} \times \mathbf{2}$ Combined Contingency Table

LHS	**RHS**	
	Y	$\overline{Y}$
X,V	All (X,V,Y) are present	Y is absent
Other	Y is present and, neither (X or V) are present or One of them is present	Y is absent and, neither (X or V) are present or One of them is present

The idea behind using combined contingency tables instead of using multi-way contingency tables, is illustrated in the two following examples;

- For three dichotomous gender (female, male), and two different diseases (breast cancer, prostate cancer). If we consider the multi-way contingency table, we will find cells like being female and having prostate cancer, and being male and having breast cancer. This could lead to unreliable results [82].

- Suppose we have three dichotomous variables gender (female or male), disease one (having *Rett* syndrome or not), and disease two (having *Alport* syndrome or not). In the contingency table, entries for cells like being female and having *Alport* syndrome, and being male and having *Rett* syndrome will be close to zero, as it is very unlikely to be female and having *Alport* syndrome and being male and having *Rett* syndrome.

In the process of Causal discovery, positive outcomes are more valuable in the Causal discovery. For instance, physicians are more interested in smoking subjects more than non-smoking subjects [82]. Considering all the cells with values close to zero will introduce

many redundancies, and subsequently unreliable results. Zhou, et al. [82] called any association identified with a positive outcomes Chi-square value $\mathcal{X}^2$ that is higher than a significance threshold $\mathcal{X}_p^2$ as positive association. If the Chi-square value $\mathcal{X}^2$ is less than a significance threshold $\mathcal{X}_p^2$, then this will form a zero association.

5.2.2 Rule discovery using partial association test

As mentioned in the previous section, positive association plays an important role in identifying a Causal relation. However, it is not easy to signify causality from correlations without human intervention [83]. Partial association test [80, 81] is a powerful statistical tool that can be used to test conditional independence of random variables when a controlled experiment is impossible. It tests the association between two random variables ***I*** and ***J***, when a third random variable ***C*** is present. When the association between ***I*** and ***J*** does not hold, given the different combinations of ***C***, we refer to this as zero partial association. Zero partial association means, either ***C*** is a common cause of both ***I*** and ***J***, or ***I*** causes ***C*** which causes ***J*** but there is no direct Causal relation between ***I*** and ***J*** [81]. Brich in [81] proved that *Mantel-Haenszel test* [80] is an optimal method for testing a partial association test against the other methods. It excludes the non-causal relationships, and only the potential causal relationships are included. Thus, it is suitable for testing the partial association of Causal discovery [3]. To test the partial association $PA(x, y, C)$ where ***x***, and ***y*** are two dichotomous random variables, and $\boldsymbol{C} = \{c_1, c_2 \dots c_t\}$ we use the following equation: $PA(x, y, C) = \frac{\left(\left|\sum_C \frac{n_{11c} \times n_{22c} - n_{12c} \times n_{21c}}{n}\right| - \frac{1}{2}\right)^2}{\sum_C \frac{r_{1c} \times c_{1c} \times r_{2c} \times r_{2c}}{n_c^2 (n_c - 1)}}$

Table 5.3 shows a three-way contingency table for testing the partial test. $\mathrm{n}_{11\mathrm{c}}$ represents the number of transactions in which both x and y are present together given the different combinations of C, $\mathrm{n}_{12\mathrm{c}}$ represents the number of transactions in which only $\boldsymbol{x}$ is present given the different combinations of C, n_{21c} represents the number of transactions in which only $\boldsymbol{y}$ is present given the different combinations of C, n_{22c} represents the number of transactions in which neither $\boldsymbol{x}$ nor $\boldsymbol{y}$ are present given the various combinations of C.

Table 5.3: Contingency Table for Partial Association Test

C			
LHS	**RHS**		**Total**
	Y	$\overline{\boldsymbol{Y}}$	
X	n_{11c}	n_{12c}	$r_{1c} = n_{11c} + n_{12c}$
$\overline{\boldsymbol{X}}$	n_{21c}	n_{22c}	$r_{2c} = n_{21c} + n_{22c}$
Total	$c_{1c} = n_{11c} + n_{21c}$	$c_{2c} = n_{12c} + n_{22c}$	$n_c = n_{11c} + n_{12c} + n_{21c} + n_{22c}$

If the partial association is greater than a significance threshold, then the association between (x, y) holds (non-zero partial association), and $x \rightarrow y$ is a Causal rule [82]. Note that the number of possible combinations of C is large $(2^m - 2)$, where m is the total number of variables. The worst memory usage and run time for conducting the partial test could thus be large. For instance, with C =3 variables, there are eight 2×2 possible contingency tables for the partial test. However, instead of testing all the combinations, we only consider the items that come in the same transaction with X or Y. Those rows or columns in the contingency table with zero values are not considered [81]. Zhou, et al. [82] have proposed the CR-PA algorithm for discovering causal rules in observational data. The basic idea of the CR-PA algorithm is to identify positive associations using association rules mining as we have mentioned earlier. These positive associations are considered as

causal hypotheses rules. Then, they employed the partial association tests on these association rules to exclude non-persistent associations.

5.3 Proposed Methods:

In this section, we present our proposed algorithms for discovering both single and combined causal rules from uncertain data. Our algorithms for discovering Causal rules from uncertain itemset mining are designed to discover

- All the k-disjunctive combined Causal rules of the form: $(a_1 \wedge a_2 \wedge \cdots \wedge a_{M-T-1} \Rightarrow t_1 \vee t_2 \vee \cdots \vee t_{k-1})$.
- All the k- conjunctive combined Causal rules of the form: $(a_1 \wedge a_2 \wedge \cdots \wedge a_{M-T-1} \Rightarrow t_1 \wedge t_2 \wedge \cdots \wedge t_{k-1})$.

5.3.1 Disjunctive Association Rules Mining from Uncertain Data

In our previous work [58], we introduced disjunctive rules from uncertain data for the first time and presented an algorithm called *DRMUD*. In our *DRMUD* algorithm, we defined a *k-disjunctive rule* as*:* $a \Rightarrow b_1 \; or \; b_2 \; or \; \cdots or \; b_{k-1}$. When item a occurs with enough support and confidence, one or more of the items $b_1, b_2, \cdots, \text{and } b_{k-1}$ are also expected to occur. We defined the rule $a \Rightarrow b_1 \; or \; b_2 \; or \; \cdots or \; b_{k-1}$ to be interesting, if each of the rules $a \Rightarrow b_j, \text{for } 1 \leq j \leq (k-1)$, has enough minimum support. This is because when the rule $a \Rightarrow b_1$ has enough support, then obviously the rule $a \Rightarrow b_1 \; or \; b_2$ also will have enough support whatever the support of the rule $a \Rightarrow b_2$ is. When the rule $a \Rightarrow b_2$ does not have enough support, then the rule $a \Rightarrow b_1 \; or \; b_2$ may not be interesting even if the rule has sufficient

support. To identify interesting rules, we introduced two support thresholds $minsup1$ and $minsup2$. Here $minsup1$ is the minimum support that each of the rules $a \Rightarrow b_j$ (for 1 ≤ *j* ≤ (*k-1*)) should have, and $minsup2$ is the minimum expected support that is required between the item *a*, and the set of items $b_1\ or\ b_2\ or\ \cdots or\ b_{k-1}$, for the rule to be interesting. *DRMUD* algorithm consists of two phases. In phase1, we mine pairs of items that have at least a minimum expected support ($minsup1$). In phase2, we use these pairs to generate *k*-disjunctive rules that satisfy the minimum expected support ($minsup2$).

5.3.1.1 Disjunctive Combined Causal Rules from Uncertain Data Algorithm- (DCCRUD)

- ***Input***:
- *A Record set R with $|R| = N$,*
- *A set of variables V with $|V| = M$ in the records, that consist of a set A of attributes $\{a_1, a_2, \cdots, a_{M-T}\}$, and a set T of targets $\{t_1, t_2, \cdots, t_{M-A}\}$, with $|A| = M - T$, and $|T| = M - A$, respectively.*
- *Size k of rules, $minsup$, significance threshold $p1$ for both Chi-square and partial association test, and significance threshold $p2$ for generating the disjunctive rules, where $p2 > p1$.*
- *Let $esup$ be the expected support.*
- *Let PA and ZA stand for all positive associations ($\mathcal{X}^2_{a_i,t_i,} > \mathcal{X}^2_{p1}$), and all zero associations ($\mathcal{X}^2_{a_i,t_i,} < \mathcal{X}^2_{p1}$), respectively.*
- *Let CR stand for both single and combined causal attributes with targets.*

Output: *All the k-disjunctive combined causal rules of the form:*

- *($a_1 \wedge a_2 \wedge \cdots \wedge a_{M-T-1} \Rightarrow t_1 \vee t_2 \vee \cdots \vee t_{k-1}$).*

Algorithm:

Phase 1: Pre-Processing

- *If the dataset is in a horizontal format, convert it to a vertical format. A vertical format is given by a set of variables each associated with a set of records the variable appears in.*
- *Delete infrequent single attributes.*

Phase 2: Find the set of positive association and zero association pairs

- *For each attribute with each target ∈ V do:*
- *Find out the set of uncertain frequent pairs (a_i, t_i) using the $F^1 \times F^1$ method.*
- *Create the contingency table for candidate pairs (a_i, t_i), and calculate the Chi-square $\mathcal{X}^2_{a_i,t_i,}$ for the contingency table.*
- *If $\mathcal{X}^2_{a_i,t_i,} > \mathcal{X}^2_{p1}$, then, insert ($a_i$, t_i) into PA,*
- *Otherwise, insert (a_i, t_i) into ZA.*

Phase 3: Find the set of Causal pairs

- *For each (a_i, t_i) ∈ PA do:*
- *Create the contingency table for partial test $PA(a_i, t_i, C)$ and calculate its partial association.*
- *If $PA(a_i, t_i, C) > \mathcal{X}^2_{p1}$, then, insert ($a_i$, t_i) into CR*
-

Phase 4: Find the set of combined Causal pairs

- *For each (a_i, t_i) ∈ ZA do:*

/ we generate combined attributes by merging two attributes using the Fk-1 × F1 method. Let A_1 ={ a_1, a_2,…, a_{k-1}} be a set of (k-1)- attributes that are associated with a certain target **t**, where each $A_1 \rightarrow t$ is a causal rule, and A_2={a_1} is a 1- attribute that is associated with the same target **t**, and $A_2 \rightarrow t$ is a causal rule. We keep $a = \{A_1 \cup A_2\}$ ➔ t as a candidate causal rule. */*

- *For each combined attribute with the same target, we generate both positive and zero associations similar to phase 2.*
- *Each positive association is tested for causality similar to phase 3.*

- *For the zero associations, we repeat the process until all the combined causal rules are found.*
- *Return CR.*

Phase 5: Generate the k-disjunctive combined causal rules

For each $a \in CR$ *do:*

For every $(k-1)$ *subset* $S \ of \ CR - \{a\}$ *do*

Let $S = \{t_1, t_2, \cdots, t_{k-1}\}$ *for specific* a

Let PAValue=0

for $1 \leq j \leq k-1$ *do*

$PAValue \ +=^{*} PA(a, t_j)$

endFor

if $PAValue \geq \mathcal{X}^2_{p2}$

Output $a_1 \wedge \cdots \wedge a_{M-T-1} \Rightarrow t_1 \vee \cdots \vee t_{k-1}$

endIf

endFor

endFor

- *Note*:* $PA(a, t_j)$ *stores the result of the partial test for each* (a, t_j).

Note $\mathcal{X}^2_{p1}$ is the p-value that each of the rules $a \Rightarrow b_j$ (for $1 \leq j \leq (k-1)$) should have to be a causal rule, and $\mathcal{X}^2_{p2}$ is the p-value that is required between the item *a*, and the set of items $b_1 \ or \ b_2 \ or \ \cdots or \ b_{k-1}$, for the rule to be interesting causal rule.

5.3.2 Conjunctive Association Rules Mining from Uncertain Data

We proposed algorithm conjunctive combined causal rules from uncertain data - *CCCRUD* algorithm.

5.3.2.1 Conjunctive Combined Causal Rules from Uncertain Data Algorithm- (CCCRUD)

Input:

- A Record set R with $|R| = N$,
- A set of variables V with $|V| = m$ in the records, that consist of two categories;
  - a set A of attributes $\{a_1, a_2, \cdots, a_q\}$.
  - a set T of targets $\{t_1, t_2, \cdots, t_{m-q}\}$,. Please note that each rule will have a subset of *V* in the precedent of the rule and the remaining attributes of *V* in the consequent. *A* corresponds to the precedent and T corresponds to the consequent.
- $minsup$ is the minimum support, significance threshold is p, and $esup$ is the expected support.
- Let PA be a set of all positive associations attributes associated with targets,
- Let ZA be a set of all zero association attributes not associated with targets.
- Let CR be a set of all Causal rules between attributes and targets.

Output: All the conjunctive combined Causal rules of the form:

$(a_1 \wedge a_2 \wedge \cdots \wedge a_q \Rightarrow t_1 \wedge t_2 \wedge \cdots \wedge t_{m-\mathrm{q}})$.

Algorithm:

Phase 1: Pre-Processing

1. Convert the dataset from a horizontal format to a vertical format, such that each variable is associated with a set of records that the variable is present in.
2. Scan through the database, and delete all infrequent 1- attributes (single attributes) that have an expected support less than $minsup$.

Phase 2: Find the set of Causal Pairs

For every $a \in A$ do

For every $t \in T$ do

/*Find out the set of all positive association pairs (2-variables (a_i, t_i) */

if a and t are in at least one record together, calculate $esup\ (a, t)$

if $(esup\ (a, t) \geq minsup)$, create contingency table, and calculate Chi-square $\mathcal{X}^2_{a,t,}$ for the contingency table.

- *If* $\mathcal{X}^2_{a,t,} > \mathcal{X}^2_p$, then, insert (a, t) into *PA,*
- Otherwise, insert (a, t) into *ZA.*

end for

end for

For each positive association pair $(a_i, t_i) \in PA$ do*:*

-Create contingency table for each *partial test,* $PA(a, t, C)$.

-*If* $PA(a, t, C) > \mathcal{X}^2_p$, then, insert (a, t) into *CR*

end for

Phase 3: Find the set of conjunctive Causal rules

For every Causal pair rule $(a_i, t_i) \in$ CR do:

/* Generate combined targets using the $F^{k-1} \times F^1$ method by merging two targets in ascending order of variable IDs.

$T_1 = \{t_1, t_2, \ldots, t_{k-1}\}$ is a (k-1)- target associated with one attribute $\boldsymbol{a}$ as Causal rule, and $T_2 = \{t_1\}$ a 1- target associated with the same attribute $\boldsymbol{a}$ as Causal rule, to generate a➔t=$\{T_1 \cup T_2\}$ a candidate causal rule with k-targets. */

For every $t_1 \in t_i$ associated with one attribute $\boldsymbol{a}$ do

For every $t_2 \in t_i$ associated with the same attribute $\boldsymbol{a}$ do, and $t_1 \neq t_2$

if t_1 and t_2 are in at least one record together, calculate expected support for attribute ***a*** and the new combined target $t = t_1 \cup t_2$

if $(esup\ (a,t)\ \geq\ minsup)$, Create the contingency table, and calculate the Chi-square $\mathcal{X}^2_{a,t}$ for the contingency table.

- ***If*** $\mathcal{X}^2_{a,t}\ >\ \mathcal{X}^2_{p}$, then, insert (a,t) into PA,
- Otherwise, insert (a, t) into ZA.

end for

end for

For each positive association pairs $(a, t) \in$ PA do:

-Create the contingency table for each partial test, $PA(a, t, C)$.

-***If*** $PA(a, t, C)\ >\ \mathcal{X}^2_{p}$, then, insert (a, t) into CR

end for

-Repeat until all the combination of combined targets Causal rules are found.

end for

Phase 4: Find the set of combined Causal Rules

For each $(a_i, t_i) \in$ ZA do:

/* Generate combined attributes using $F^{k-1} \times F^1$ by merging two attributes in ascending order of variable IDs

$A_1 = \{a_1, a_2, \ldots, a_{k-1}\}$ is a (k-1)- attributes associated with one target ***t*** as Causal rule, and

$A_2 = \{a_1\}$ a 1- attribute associated with the same target ***t*** as Causal rule, to generate $a = \{A_1 \cup A_2\}$ ➔ t a candidate causal rule with k-attributes. */

For every $a_1 \in a_i$ associated with one target ***t*** do

For every $a_2 \in a_i$ associated with the same target ***t***, and $a_1 \neq a_2$ do

if $\mathrm{a}_1$and a_2 are in at least one record together, calculate expected support for one target ***t*** and the new combined attributes $\mathrm{a} = \mathrm{a}_1 \cup \mathrm{a}_2$

if $(esup\ (a,t)\ \geq\ minsup)$, Create the contingency table, and calculate the Chi-square $\mathcal{X}^2_{\mathrm{a,t}}$ for the contingency table.

- ***If*** $\mathcal{X}^2_{\mathrm{a,t}}\ >\ \mathcal{X}^2_{\mathrm{p}}$, then, insert (a,t) into PA,
- Otherwise, insert (a, t) into ZA.

end for

end for

For each positive association pairs $(\mathrm{a}, \mathrm{t}) \in$ PA do:

-Create the contingency table for each partial test, $\mathrm{PA}(\mathrm{a}, \mathrm{t}, \mathrm{C})$.

-***If*** $\mathrm{PA}(\mathrm{a}, \mathrm{t}, \mathrm{C})\ >\ \mathcal{X}^2_{\mathrm{p}}$, then, insert (a, t) into CR

end for

-Repeat until all the combination of combined attributes Causal rules are found.

end for

5.4 Complexity Analysis

To compute the complexity for both $DCCRUD$ and $CCCRUD$ algorithms, consider the following parameters;

- n = the number of records,
- q = average number of variables in a record,
- m = total number of variables,
- ρ = average size of each item list
- V^k= a generic *k*- tuple of variables,

- $R(V^k)$ = a list of records that V^kappears in,
- F^k= set of all frequent *k*-tuple of variables.

5.4.1 *DCCRUD* Complexity Analysis

The *DCCRUD* algorithm finds all the single and combined causal rules under a given minimum support $minsup$, and two significance level thresholds (*p1, p2)*. Here *p1* stands for both Chi-square and partial association test, and significance threshold *p2* stands for generating the disjunctive rules. *DCCRUD* starts by converting the records in horizontal layout to vertical layout in a single scan. The vertical layout views the data as a list of records, one for each variable. The list corresponding to any variable is the list of records in which the variable occurs.

That is in phase 2 the algorithm generates frequent pairs using the $\mathrm{F}^1 \times \mathrm{F}^1$method by merging a 1-frequent variables with a 1-frequent variables in ascending order of item IDs to generate a 2-frequent variables, and this can be done in linear time. Also, *DCCRUD* counts the expected support for a new 2-variable itemset by making intersection between record lists. Due to ordering of the records this intersection can be done in time that is linear in the total length of the two lists. Note, we don't delete the LHS of the infrequent pairs, but they are excluded from consideration for combinations with other variables in the partial test. For example, let the set of attribute variables $\mathrm{be}\ \{1,2,..8\}$, and we have two targets $\{9,10\}$. If a pair (1=LHS/attribute, 10=RHS/target) is infrequent, we can't delete the attribute '1', because it might be frequent and positively associated with another target. For example, the pair (1=LHS/attribute, 9= RHS/target) could be frequent and positively associated. Also, when the pair (2=LHS/attribute, 10= RHS/target) is frequent and

positively associated, then we exclude the variable 1 to be tested in the partial test, only for target 10. Since we already did count the expected support for n_{11} in the previous step, we can easily infer the other values using the following formulas:

$\mathrm{n}_{11} = esup\ (a_\mathrm{i}, t_\mathrm{i}),\quad \mathrm{n}_{12} = esup\ (a_\mathrm{i}) - \mathrm{n}_{11}$

$\mathrm{n}_{21} = esup\ (t_\mathrm{i}) - \mathrm{n}_{11},\ \mathrm{n}_{22} = R - esup\ (a_\mathrm{i}) - esup\ (t_\mathrm{i}) + \mathrm{n}_{11}$

DCCRUD stores all the pairs $(a_\mathrm{i}, t_\mathrm{i})$ that pass the chi-square test in PA as positive association values. Otherwise, it stores them in ZA as zero association values.

In phase 3: *DCCRUD* creates the 2×2 contingency tables for each positive association pairs $(a_\mathrm{i}, t_\mathrm{i})$ with all combinations of C (all variables that are already associated with $(a_\mathrm{i}, or\ t_\mathrm{i})$ and calculate the partial association test, using *Mantel-Haenszel partial association test* (see table 5.3).

To generate combined rules from the zero associations, in phase 4, we adopt the same strategy in [82]. The rule $a_1 \wedge a_2 \rightarrow t_\mathrm{i}$ can be combined if both $(a_1 \rightarrow t_\mathrm{i})$ have a zero association and $(a_2 \rightarrow t_\mathrm{i})$ have a zero association. This will reduce the search space since we will exclude all the positive associations from further combining. To determine the Chi-Combined causal rules, all exposure values (LHS=1) are combined into one variable, and all non-exposure values (LHS=0) are combined together (see table 5.2). Based on observations in [62, 82], if the rule $(a_1 \rightarrow \mathrm{t})$ is an association rule but it fails in the partial test to be a causal rule, this means, the association between $\boldsymbol{a_1}$ and $\mathbf{t}$ is either interrupted by other variables, or the other variables are a common cause of both $\mathbf{a_1}$ and $\mathbf{t}$. Clearly, there is no direct causal relation between $\mathbf{a_1}$ and $\mathbf{t}$**.** Thus, it is improbable that combining another variable with the LHS will lead to a direct association. Moreover, any superset of a causal rule is considered as a redundant rule, and doesn't give any new information

[82,83]. This will reduce the search space since once a causal rule is discovered we will not generate any superset of this rule. Also, the regular frequent association itemsets serve as pruning conditions that reduce the search space and time.

Following the same assumption of our proposed algorithm on disjunctive rule mining (DRMUD algorithm [58]), given a chi-square calculation for two causal rules (i.e. $x \rightarrow a_1$, $x \rightarrow a_2$), the rule $x \rightarrow a_1 \vee a_2$, must have a significance threshold $\mathcal{X}_{p2}^2$ and each rule $x \Rightarrow a_j$ must have a significance threshold $\mathcal{X}_{p1}^2$ (for 1 ≤ *j* ≤ (*k-1*)), where $\mathcal{X}_{p2}^2 > \mathcal{X}_{p1}^2$.

In Phase 5, while considering the set $S = \{t_1, t_2, \cdots, t_{k-1}\}$, if there is any j $(1 \leq j \leq k-1)$ such that PAValue $(a_i, t_j) < \mathcal{X}_{p1}^2$. , then the set *S* is not considered.

In summary, the time complexity for our algorithm is as follows.

Phase 1: the first step takes $O(q * n)$ time. Each variable is allocated a bucket indexed with the same id as the variable. In a single scan through the database we can figure out $R(V^k)$ for all 1-Varaibles. The second step takes time $O(m)$, for only a single scan through the set of all 1-Varaibles. **The time complexity** for this phase is $O(q * n + m)$.

Phase 2 takes $O(1)$ time for a 1-frequent itemset. Total time for all the 1 itemsets in the set is given by $\binom{m}{2} p$. Testing all pairs requires an expected time of $\binom{m}{k} p$. **The time complexity** for this phase is $O\left(\binom{m}{k} p + p \sum_{k=1}^{m} \binom{m}{k}^2\right)$

Phase 3, and **Phase 4 a**re the most crucial in the algorithm because these phases are iterated. **The time complexity** for these phases are $O\left(\binom{m}{k} p\right)$

Phase 5: takes time $O\left(mk \binom{m-1}{k-1}\right) = O(km^k)$.

5.4.2 *CCCRUD* Complexity Analysis

CCCRUD algorithm first, converts records from horizontal format to vertical format, in which each variable is associated with a set of records the variable is present in. The vertical layout outperforms the horizontal layout in mining frequent patterns [55]. This step takes $O(q * n)$ time. Since each variable is allocated a bucket indexed with the same id as the variable, we can figure out $R(V^k)$ for all 1-Varaibles in a single scan through the database. *CCCRUD* then, deletes all infrequent variables with expected support less than the minimum support in only a single scan through the set of all 1-Varaibles. The step takes $O(m)$ time. **The time complexity** for this phase $O(q * n + m)$.

In the second phase, *CCCRUD* generates all the causal pairs, such that, the pair (attribute, target) has to be frequent, positively associated, and nonzero partial associated. In other words, the, pair (attribute, target) must have an expected support of at least *minsup*, its Chi-square $\mathcal{X}^2_{a,t,}$ has to be greater than the significant threshold, and it should pass the partial test with a value also greater than the significant threshold. In this phase, we don't delete the attributes of infrequent pairs since any of these attributes can form frequent pairs with a different target. Instead, we don't consider the attribute in the infrequent pair (attribute, target) as one of the combination variables for testing the partial test. For instance, suppose $A = \{1,2,..8\}$ is the set of attribute variables, and $T = \{9,10\}$ is the set of targets. Let (1, 10) be an infrequent pair, then the attribute '1' will not be considered when we test the partial test for any positive association pair with 10 as a target (i.e., (2,10), (3,10), …, (8,10)). For each frequent pair (attribute, target), we calculate the observed values for the four cells on the contingency table (see table 5.1), by only counting the expected support for n_{11}, and inferring the other values using the following formulas:

$$\mathrm{n}_{11} = esup\,(a_\mathrm{i}, t_\mathrm{i}),\quad \mathrm{n}_{12} = esup\,(a_\mathrm{i}) - \mathrm{n}_{11},$$

$$\mathrm{n}_{21} = esup\,(t_\mathrm{i}) - \mathrm{n}_{11},\ \mathrm{n}_{22} = R - esup\,(a_\mathrm{i}) - esup\,(t_\mathrm{i}) + \mathrm{n}_{11}$$

All positive association pairs (attribute, target) in *PA* that pass the chi-square test will be tested using the Mantel-Haenszel partial association test. Otherwise CCCRUD stores them in *ZA* as zero association values. All positive association pairs (attribute, target) that pass the Mantel-Haenszel partial association test will be stored in *CR*. **The time complexity** for this phase is $O\left(\binom{m}{k} p + p\sum_{k=1}^{m}\binom{m}{k}^2\right)$

Where the first step, generating the frequent pairs using the $\mathrm{F}^1 \times \mathrm{F}^1$ method cost linear time in the total length of the two lists due to ordering of records. It takes $O(1)$ time for a 1-frequent itemset. Total time for all the 1 itemsets in the set is given by $O\left(\binom{m}{2} p\right)$, and to test all the pairs that requires an expected time of $O\left(\binom{m}{k} p\right)$Then, for each positive association pair (attribute, target) in *PA*, testing partial test for Causal relation with all possible combinations takes $O\left(\binom{m}{k} p\right)$ time, and all positive association pairs takes $O\left(p\sum_{k=1}^{m}\binom{m}{k}^2\right)$) time.

In the third phase, only causal pairs in *CR* will be used to generate conjunctive causal rules. Here conjunctive causal candidates are nothing but combined targets. We merge two targets in ascending order of variable IDs using the $\mathrm{F}^{k-1} \times \mathrm{F}^1$ method. Similar to phase two, we generate all the conjunctive causal rules. **The time complexity** for this phase is $O\left(\binom{m}{k} p + p\sum_{k=1}^{m}\binom{m}{k}^2\right)$

In the last phase, we generate combined rules from the zero associations sorted in *ZA*. Like in [82], we have combined two zero association rules $(a_1 \rightarrow t_\mathrm{i})$ and $(a_2 \rightarrow t_\mathrm{i})$ into $a_1 \wedge$

$a_2 \rightarrow t_{\mathrm{i}}$ to check if the new combined rule can form a positive association which might lead to a causal rule. Here the combined causal candidates are obtained using the $F^{k-1} \times F^1$ method. Since we have only combined the zero associations, the search space will be reduced. Another observation is adapted from [62, 82] to reduce time as follows: adding more attributes to a positive association rule that is not causal, will not lead to a direct association. Furthermore, the traditional downward closure property is used as a pruning condition that reduces the search space and time. Also, the upward closure property is used. That is, any superset of a causal rule is a redundant rule that doesn't give any new information [82, 83]. **The time complexity** for this phase is $O\left(\binom{m}{k} p + p \sum_{k=1}^{m} \binom{m}{k}^2\right)$

5.5 EXPERIMENTAL RESULTS

Before we discuss the results of our proposed algorithm, we show the comparison between our proposed algorithm and the algorithm in [82] for single target under certain database.

5.5.1 CCRCD algorithm V.s. CR-PA algorithm

Our Combined Causal Rule from Certain Data (*CCRCD*) algorithm has discovered the same causal rules as the *CR-PA* algorithm [82] for the case of single targets. We have used two datasets. We downloaded dataset1 from [84]. This dataset has also been used in [62] for discovering causal rules using *PC* [86], *HITON-PC* [87] and *CR-PA* [82] algorithms with eight attribute variables and one target variable, within 100k records. Dataset 2 has been generated using the software tool mentioned in [85] with 29 attribute variables and one target variable within 856 records.

Table 5.4 shows the results for both *CR-PA* and *CCRCD* algorithms under dataset 1 and minimum support=0.05, and p-value= 0.05 (95%). Note that in the original data the names for the attributes were (A, B, C, D, E, F, G, H), and the target name was (Z). We changed the attribute names to the values {1, 2…, 8}, and the target to the value {9}. Table 5.5 shows the results for both *CR-PA* and *CCRCD* algorithms under dataset 2 with minimum support=0.01, and p-value= 0.05 (95%). Here the attributes names are {1, 2…, 29}, and the target name is {30}. Table 5.6 shows the results for both *CR-PA* and *CCRCD* algorithms under dataset 2 with minimum support=0.01, and p-value= 0.1 (90%).

Table 5.4: Dataset 1- Causal Rules under *p=0.05, min. support=0.05*

CR-PA	***CCRCD***
B→ Z	2→ 9
C→Z	3→9
F→Z	6→9

Table 5.5: Dataset 2 - Causal Rules under *p=0.05, min. support=0.01*

Causal rule	***CR-PA***	***CCRCD***
3→ 30	√	√
6 → 30	√	√
22→ 30	√	√
23→ 30	√	√
27→ 30	√	√
29→ 30	√	√
11∧28→ 30	√	√

Table 5.6: Dataset 2- Causal Rules under *p=0.1, min. support=0.01*

Causal rule	***CR-PA***	***CCRCD***
2→ 30	√	√
3 → 30	√	√
6 → 30	√	√
15 → 30	√	√
22 → 30	√	√
23 → 30	√	√
27 → 30	√	√
29 → 30	√	√
9∧17→ 30	√	√
11∧28→ 30	√	√

The time complexity for discovering a rule like $X \rightarrow Z$ using $CR-PA$ algorithm is $O(l^2)$, where I represents different records in the database. $CCRCD$ algorithm takes $O\left(\binom{m}{k} p + p \sum_{k=1}^{m} \binom{m}{k}\right)\ time$, where $O\left(\binom{m}{k} p\right)$ time for testing each pair ($X \rightarrow Z$) if they form a positive association rule, and $O\left(p \sum_{k=1}^{m} \binom{m}{k}\right)$ is for testing the partial test to see if the positive association rule forms a causal rule.

5.5.2 Experimental Results for *DCCUCM* Algorithm

Since there is no existing algorithm for generating disjunctive combined causal rules from uncertain data, we have combined our method -*DRMUD* algorithm [58] with *CR-PA* algorithm [82] and proposed the *DCCRUD* algorithm to mine disjunctive combined causal rules from data without uncertainty. *DCCRUD* is implemented and compiled using Java. Following is the execution environment:

- Intel(R) Core(TM) i7 3.40GHz PC
- 8GB Main memory
- Operating System is Microsoft Windows 7.

For testing the performance of *DCCRUD* over uncertain databases, we have used the following procedure:

We generated two datasets using [85]. Dataset 1 consists of 5142 records with 20 attribute variables and five target variables, and dataset 2 consists of 26380 records with 40 attribute variables and ten target variables.

For each dataset, we have run our algorithms with different size of the rules K, and w and different configuration parameters of minimum expected support (*0.001*, and 0.0009),

and *p1−value: 0.1* (90%), along with *p2−value: 0.05* (95%), and p1−value" *0.025* =97.5%, along with *p2−value: 0.01* =99%. Table 5.7 shows the results of disjunctive single and combined Causal rules under p1-value= =*0.025*, p2-value= =*0.01*, *minimum support=0.001*, with rule maximum size *k=2*.

Table 5.7: Dataset 2- Disjunctive single and combined Causal Rules DCCRUD

5 → 21	13 → 23
6 → 22 ∨ 24	14 → 21
7 → 22	19 → 23
8 → 21	4∧5 → 25
10 → 21	3∧ 16 → 21
12 → 22	17 → 22 ∨ 24

Figure 5.1 shows the performance of our proposed algorithm. Note that the runtimes can vary based on other parameters provided by the users. Figure 5.2 shows the scale-up of the attributes. The run time increases as the number of attributes increases. Also, the run time decreases as the minimum support increases as is shown in figure 5.3

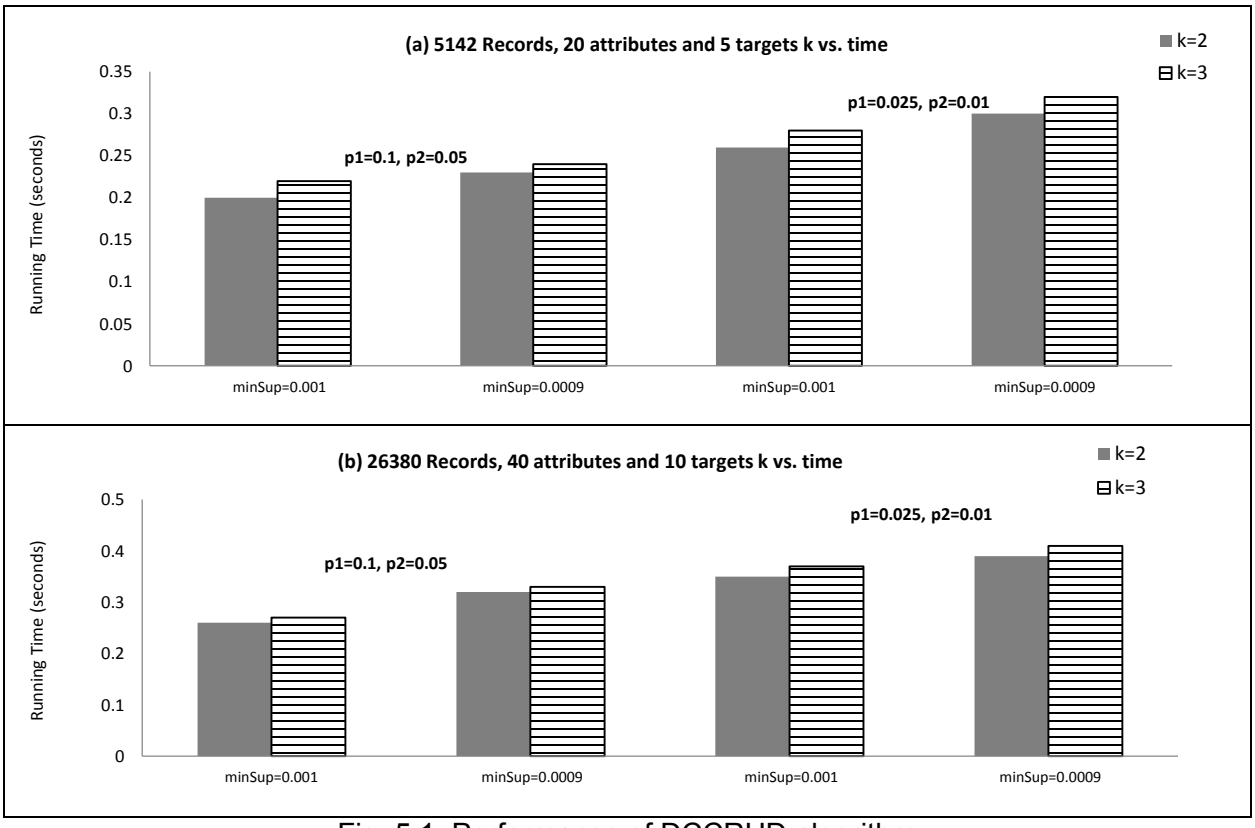


Fig. 5.1. Performance of DCCRUD algorithm

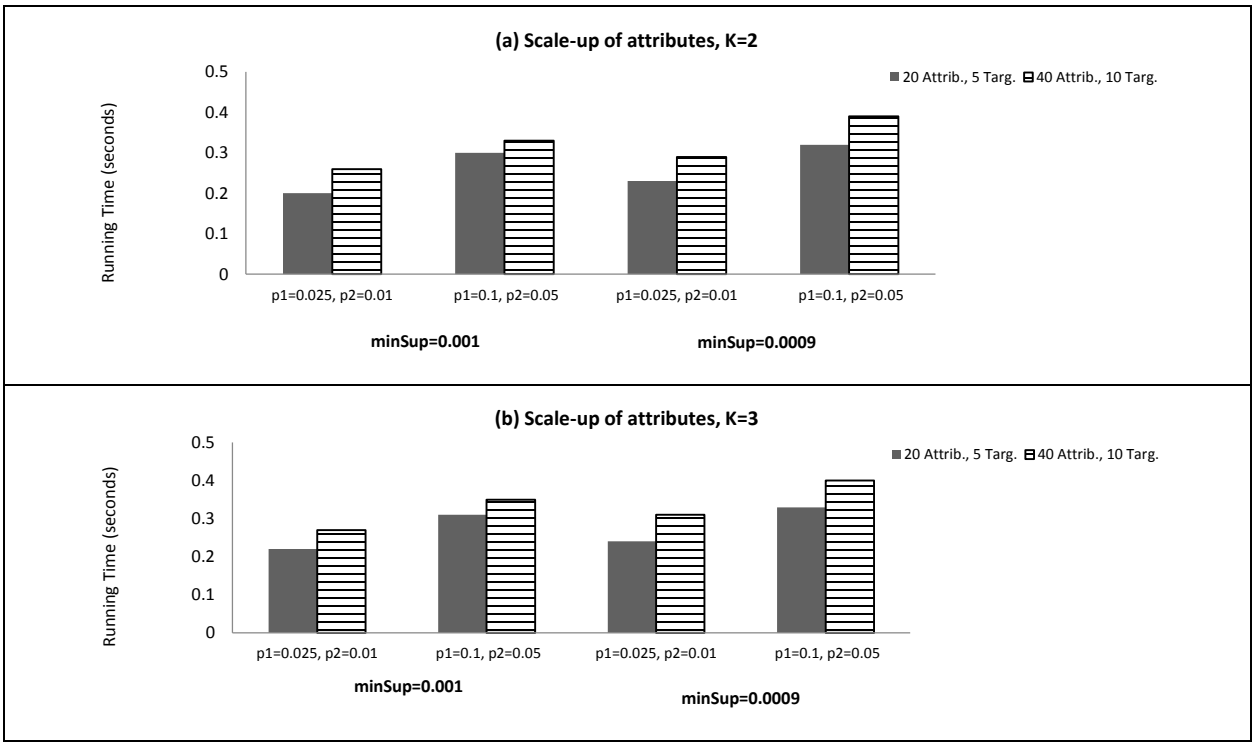


Fig.5.2. Scale-up of attributes

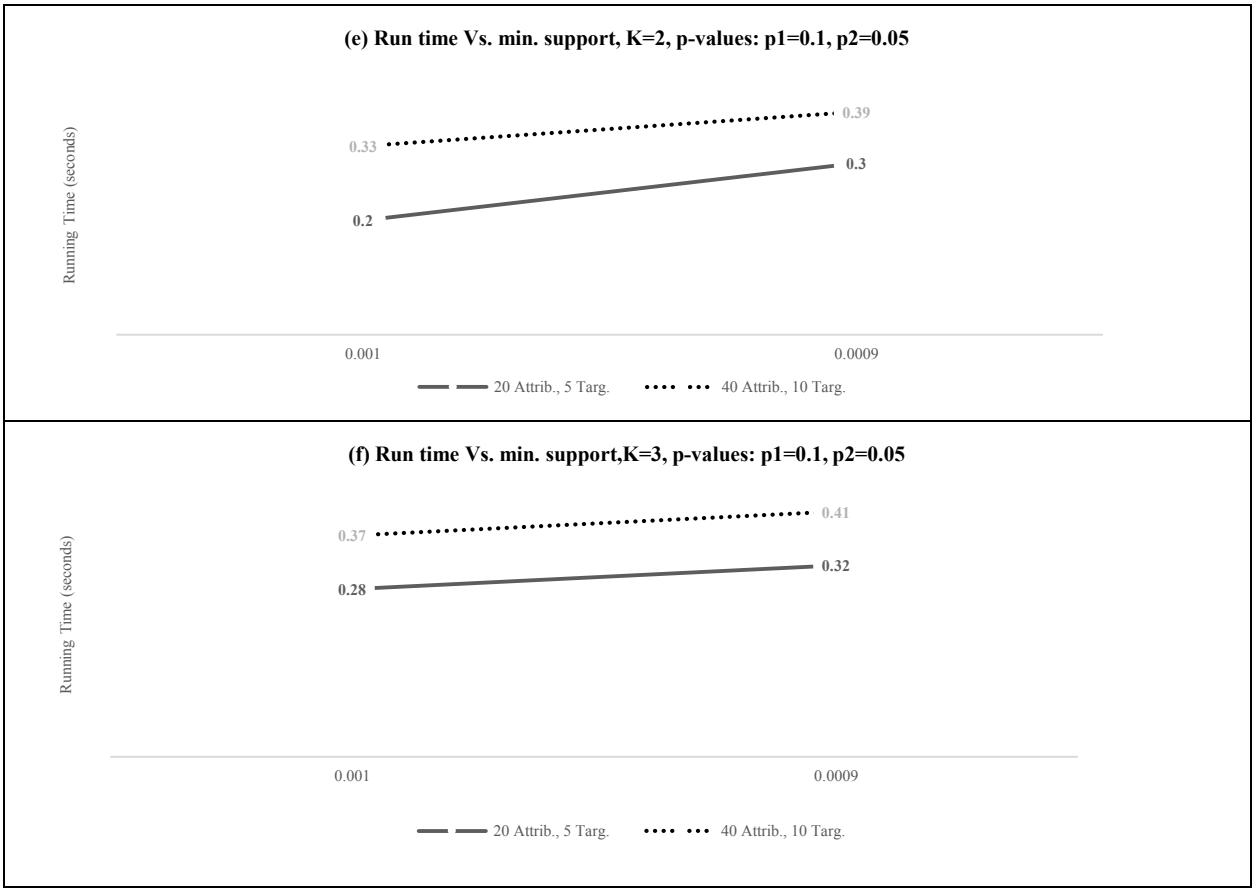


Fig. 5.3. Run time Vs. *minimum support*

5.5.3 Experimental Results for *CCCUCM* Algorithm

The *CCCRUD* algorithm has been implemented and compiled using Java, and it was run on an Intel(R) Core(TM) i7 3.40GHz PC with 8GB Main memory, operating on Microsoft Windows 7. We generated three datasets using [85] to test **CCCRUD's** performance over uncertain databases; *dataset 1* consists of 7380 records with 20 attribute variables and 5 target variables, *dataset 2* consists of 27668 records with 40 attribute variables and 10 target variables, and *dataset 3* consists of 73256 records with 100 attribute variables and ten 20 variables. We used the following configuration parameters over the generated datasets; Minimum expected support (0.001, and 0.0009), *p−value* (*0.1* (90%), *0.05* (95%)), *0.025* (97.5%), *0.01* and (99%)). Table 5.8 shows the results of conjunctive single and combined Causal rules over dataset 1. Figure 5.4 shows the performance of *CCCRUD* algorithm over the three datasets where runtimes vary based on other parameters provided by the users. Figure 5.5 shows the scale-up of the attributes where the runtimes are functions of the attributes. Also, runtimes are functions of the minimum support as is shown in figure 5.6

Table 5.8: Dataset 1- Conjunctive single and combined Causal Rules CCCRUD

1→22	6→21	10→23	14→24	18→23	20→23
1→24	6→24	10→24	15→21	18→24	20→24
2→21	6→25	11→21	15→24	18→25	20→25
2→24	7→21	11→23	15→25	18→21∧23	20→21∧23
2→25	7→23	11→24	16→24	18→23∧24	20→21∧25
3→21	7→24	12→21	16→25	18→ 21∧23∧24	20→21∧23∧25
3→23	8→21	12→23	17→21	19→21	2∧6→22
3→23	8→23	12→24	17→22	19→22	2∧6→23
3→25	8→24	12→25	17→23	19→23	2∧6→22∧23
3→23∧25	8→25	13→21	17→24	19→25	4∧5→25
4→21	9→21	13→23	17→25	19→23∧24	
4→24	9→23	13→24	17→22∧23	19→23∧25	
5→21	9→24	13→25	17→22∧24	19→24∧25	
5→23	9→25	14→21	17→23∧24	20→21	
5→24	10→21	14→23	18→21	20→22	

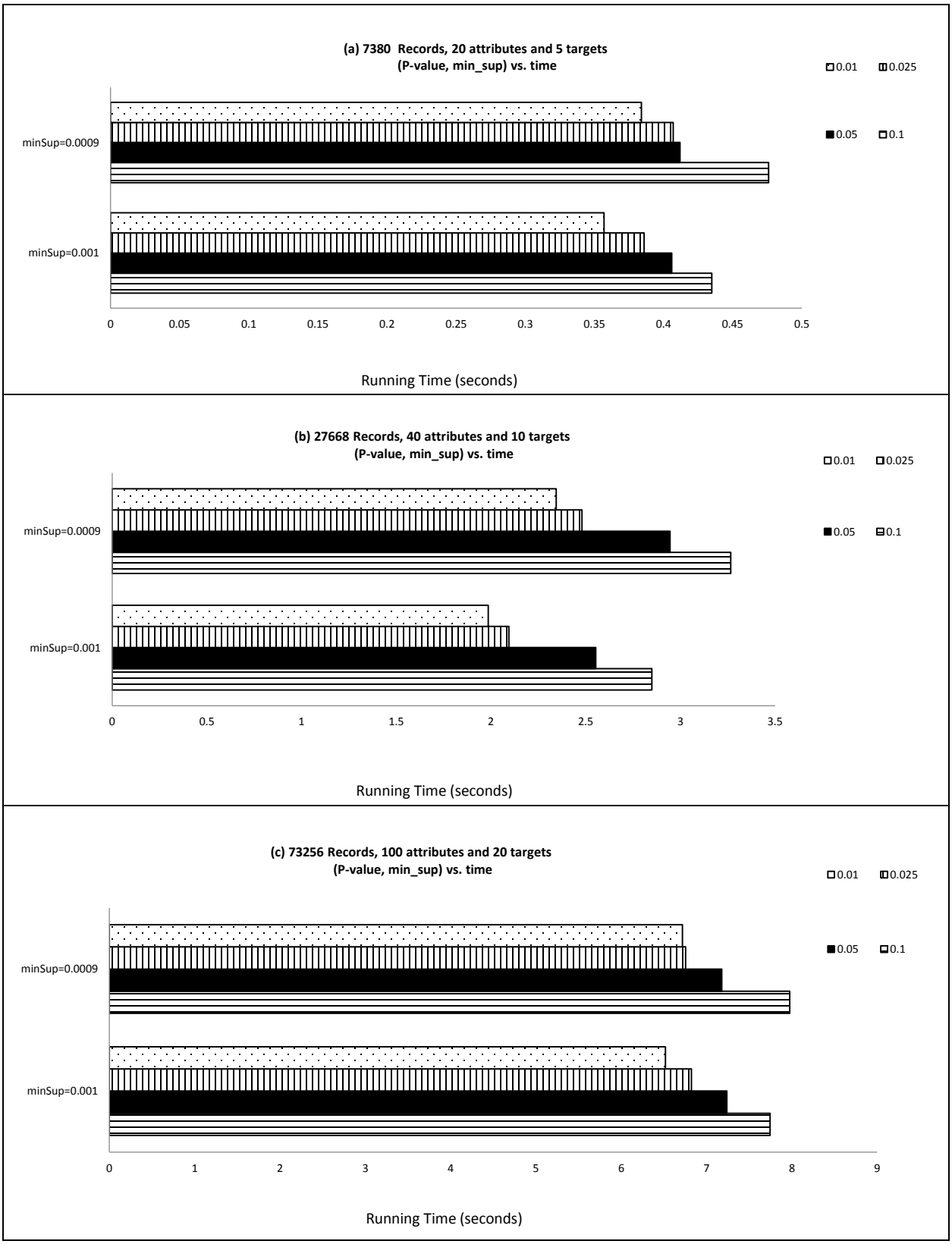


Fig. 5.4. Performance of $CCCRUD$ algorithm

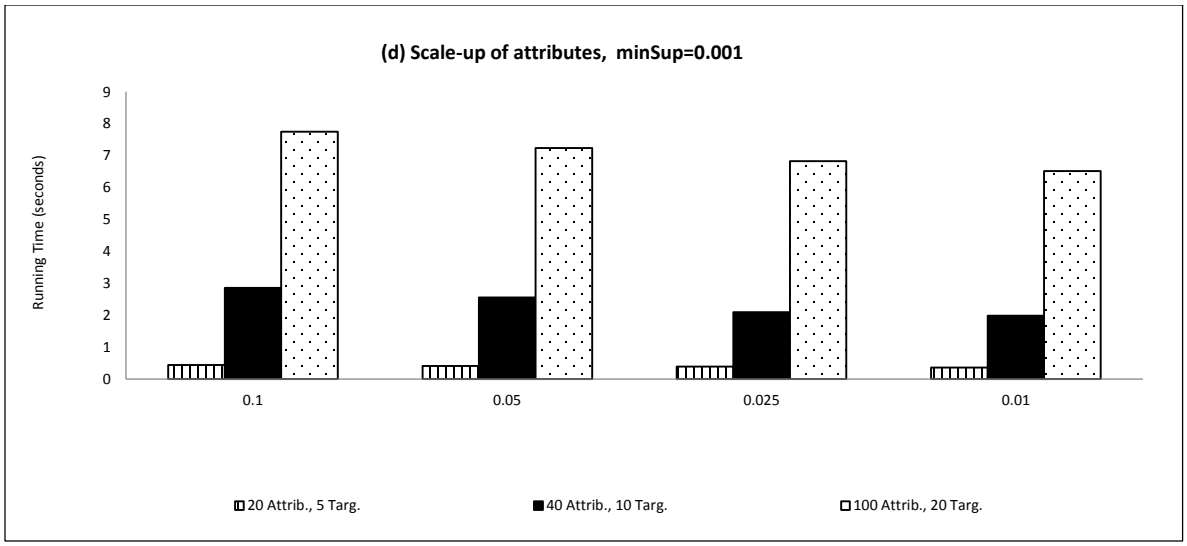


Fig. 5.5. Scale-up of attributes

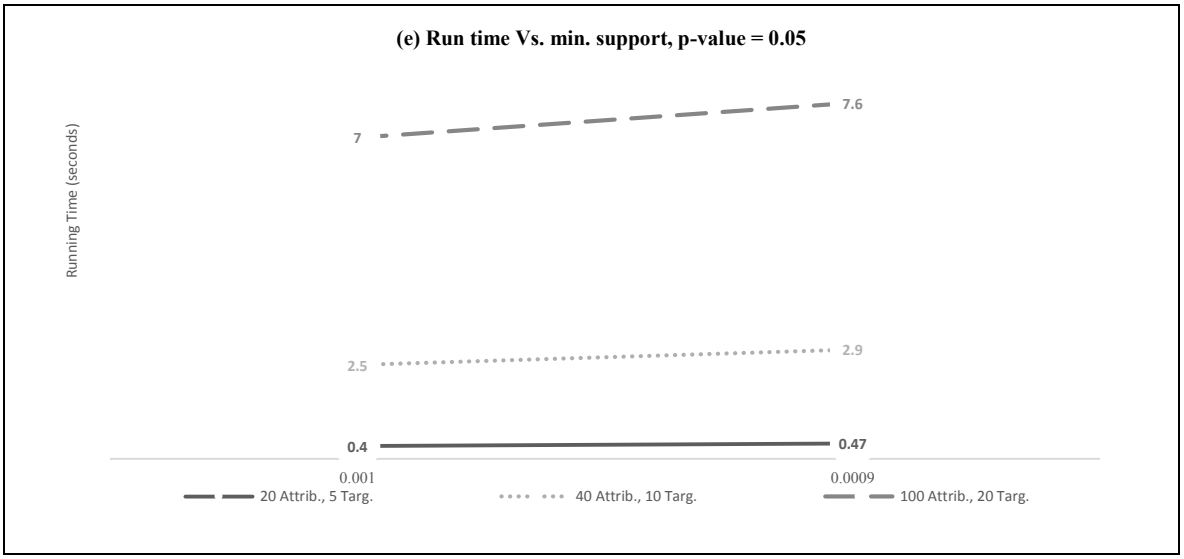


Fig. 5.6. Run time Vs. $minimum\ support$

5.6 Conclusions

Causal discovery is a well-known problem that also has been addressed using association rules mining under certain data. In this work, we extend our prior work [55] for mining conjunctive rules from uncertain data to generate conjunctive single and combined causal rules. Our proposed algorithm, namely, CCCRUD takes advantage of vertical

layout to accelerate the process of generating conjunctive rules. We believe that this is the first work that mines single and combined causal rules from uncertain data. In this paper we present both theoretical and experimental results.

In our prior work [58] we were the first to introduce disjunctive rules mining from uncertain data. In this research we have introduced and studied the problem of mining disjunctive combined causal from uncertain data. To the best of our knowledge we are the first ones to introduce this problem. We have proposed the *DCCRUD* algorithm to tackle this problem. Our algorithm *DCCRUD* works under vertical layout. We have presented theoretical and experimental results for the proposed method.

Chapter 6

Concluding Remarks and Future Directions

In this dissertation, we have presented a survey on association rules mining techniques under both certain and uncertain data. Both sequential and parallel techniques have been considered. We have identified the differences between the two approaches of mining association rules: the level-wise approach and the FP-tree growth approach. In addition, we have summarized the state-of-the-art techniques for optimizing the complexity. Our contributions in this dissertation are:

- **Weighted frequent patterns mining:** we have studied the problem of mining from uncertain weighted data and we are the first to introduce theoretical and experimental results for mining frequent patterns from uncertain weighted data. We have proposed two algorithms for this problem under two different layouts of databases: horizontal and vertical databases layouts.

- **Disjunctive association rules mining:** we **have** filled a crucial gap in rules mining. We are the first ones to introduce a new algorithm to mine disjunctive rules from uncertain data.

- **Causal rules mining**: We have extended our prior works for mining both disjunctive and conjunctive rules from uncertain data. Specifically, we have proposed two methods; one for generating disjunctive single and combined causal rules, and the other for generating conjunctive single and combined causal rules. For the two methods, we have used the vertical layout to accelerate the process of generating the disjunctive and conjunctive rules from uncertain data. We believe that this is the first work that mines single and combined causal rules from uncertain data.

In the context of frequent patterns mining under uncertain data, all the known extensions of the FP-growth algorithm do not perform well. It is an interesting open problem to discover efficient extensions of the FP-growth algorithm that work for uncertain data. In our future work we plan to address this problem.

Our algorithms can be extended to the problem of incremental association rules mining. We also plan to work on this.

Also, our algorithms for generating either disjunctive or causal rules can be extended to generate more propositional Logic like:

$$Newdrug \rightarrow [(se1 \wedge se2) \oplus se3] \vee se4,$$

$$Newdrug\ causes \begin{cases} either \begin{cases} \text{se1}\ and\ se2 \\ or\ \text{se3} \end{cases} \\ or\ se4 \end{cases}$$

We will work on this extension as well.

Bibliography

1. R. Agrawal, T. Imielinski, and A. N. Swami, “Mining association rules between sets of items in large databases”, Proceedings of ACM SIGMOD International Conference on Management of Data, ACM Press, Washington DC, pp.207-216, May 1993.
2. P. N. Santhosh Kumar, C. Sunil Kumar, C. Venugopal, “Improving Association Rule based Data Mining Algorithms with Agents Technology in Distributed Environment”, Proceedings of The Intl. Conf. on Information, Engineering, Management and Security 2014 [ICIEMS 2014].
3. F.-V., Philippe, “Un modèle hybride pour le support à l'apprentissage dans les domaines procéduraux et mal-définis**”,** Ph.D. Thesis, University of Quebec in Montreal, Montreal, Canada, 184 pages. 2010.
4. F.-V., Philippe, 2013-05-11, “How to auto-adjust the minimum support threshold according to the data size”. Available” http://data-mining.philippe-fournier-viger.com/how-to-auto-adjust-the-minimum-support-threshold-according-to-the-data-size/
5. M. H. Dunham, Y. Xiao, L. Gruenwald and Z. Hossain, “A Survey Of Association Rules”. International Journel of Computer TheoryAnd Engineering, vol.4, No.2 June 2003

6. Z. Qiankun, S. B. Sourav, “Association Rule Mining: A Survey”, Technical Report, Center for Advanced Information Systems (CAIS), Nanyang Technological University, Singapore, 2003.

7. M. Houtsma, A. Swami, “Set-Oriented Mining for Association Rules in Relational Databases”, Proceedings of the 11th IEEE International Conference on Data Engineering, pp. 25-34, Taipei, Taiwan, March 1995.

8. R. Agrawal, R. Srikant, “Fast Algorithms for Mining Association Rules in Large Databases”, Proceedings of the Twentieth International Conference on Very Large Databases, pp. 487-499, Santiago, Chile, 1994.

9. A. Savasere, E. Omiecinski, and S. B. Navathe, “An Efficient Algorithm for Mining Association Rules in Large Databases”, Proceedings of the 21nd International Conference on Very Large Databases, pp. 432-444, Zurich, Swizerland, 1995.

10. H. Toivonen, “Sampling Large Databases for Association Rules”, Proceedings of the 22nd International Conference on Very Large Databases, pp. 134-145, Mumbai, India, 1996.

11. J. Han, J. Pei,Y. Yin. “Mining Frequent Patterns without Candidate Generation”. Proc of ACM-SIGMOD, 2000.

12. J. Han, J. Pei, “Mining frequent patterns by pattern-growth: methodology and implications”, ACM SIGKDD Explorations Newsletter 2, 2, 14-20.

13. R. Agrawal, J. C. Shafer, “Parallel Mining of Association Rules”, IEEE Transactions on Knowledge and Data Engineering, Vol. 8, No. 6, pp. 962-969, December 1996.

14. J. S. Park, M.-S. Chen, and P. S. Yu, “Efficient Parallel Data Mining for Association Rules”, Proceedings of the International Conference on Information and Knowledge Management, pp. 31-36, Baltimore, Maryland, 22-25 May 1995.

15. M. J. Zaki, M. Ogihara, S. Parthasarathy, and W. Li, “Parallel Data Mining for Association Rules on Shared-Memory Multiprocessors”, Technical Report TR 618, University of Rochester, Computer Science Department, May 1996.

16. D. W.-L. Cheung, J. Han, V. Ng, A. W.-C. Fu, and Y. Fu,” A Fast Distributed Algorithm for Mining Association Rules”, Proceedings of PDIS, 1996.

17. T. Shintani and M. Kitsuregawa, “Hash Based Parallel Algorithms for Mining Association Rules”, Proceedings of PDIS, 1996.

18. E.-H. Han, G.e Karypis, and V. Kumar, “Scalable Parallel Data Mining For Association Rules”, Proceedings of the ACM SIGMOD Conference, pp. 277-288, 1997.

19. M. J. Zaki, S. Parthasarathy, M. Ogihara, and W. Li, “New Parallel Algorithms for Fast Discovery of Association Rules”, Data Mining and Knowledge Discovery, Vol. 1, No. 4, pp. 343-373, December 1997.

20. P. Shenoy, J. R. Haritsa, S. Sudarshan, G. Bhalotia, M. Bawa, and D. Shah, ”Turbo-charging vertical mining of large databases,” Procedings of ACM SIGMOD Intnational Conference on Management of Data, ACM Press, Dallas, Texas, pp. 22-23, May 2000,

21. R. Agrawal, H. Mannila, R. Srikant, H. Toivonen and A. I. Verkamo: "Fast Discovery of Association Rules", Advances in Knowledge Discovery and Data Mining, Chapter 12, AAAI/MIT Press, 1995.

22. B. Dunkel, N. Soparkar,"Data Organization and access for Efficient Data Mining",ICDE 1995.

23. M. Song, S. Rajasekaran, “A Transaction Mapping Algorithm for Frequent Itemsets Mining”, IEEE Transactions on Knowledge and Data Engineering, pp 1– 4.

24. J. D. Holt, and S. M. Chung, “Efficient Mining of Association Rules in Text Databases”. CIKM'99, Kansas City, USA, pp. 234-242, (Nov. 1999)

25. D.W. Cheung, J. Han, V. T. Ng, C.Y Wong, “Maintenance of Discovered Association Rules in Large Databases” An Incremental Update Technique. In: Proceedings of International Conference on Data Engineering, pp. 106–114 (1996)

26. D. W. Cheung, S.D. Lee, B. Kao, “A General Incremental Technique for Maintaining Discovered Association Rules” In: Proc. of the 5th International Conference on Database Systems for Advanced Applications, pp. 185–194 (1997)

27. S. Lee, D. Cheung, “Maintenance of Discovered association rules. When to update?”, In Proc. of Research Issues on Data Mining and Knowledge Discovery, pp 51- 58.

28. N.F. Ayn, A.U. Tansel, E. Arun, “An efficient algorithm to update large itemsets with early pruning”, Technical Report BU-CEIS-9908, Department of CEIS Bilkent University, June 1999.

29. C.H. Lee, C.R. Lin, M. S. Chen, “Sliding-Window Filtering: An Efficient Algorithm for Incremental Mining”, In Proceedings of the 10th International Conference on Information and Knowledge Management, Atlanta, Georgia, USA, November 2001, pp. 263–270.

30. C.C. Chang, Y.C. Li, J.S. Lee,. “An efficient algorithm for incremental mining of association rules”; Proceedings of the 15th international workshop on research issues in data engineering: stream data mining and applications (RIDE-SDMA’05), IEEE, 2005.

31. T. P. Hong, C. Y. Wang, Y. H., Tao, “Incremental Data Mining Based on Two Support Thresholds”, Proceedings of the 4th International Conference on Knowledge-Based Intelligent Engineering Systems and Allied Technologies, pp.436-439, 2000.

32. T. P. Le, T. P. Hong, B. Vo, B. Le, “ Incremental mining frequent itemsets based on the trie structure and the pre-large itemsets” Granular Computing (GrC), 2011 IEEE International Conference on 8-10 Nov. 2011

33. T.P. Hong, C.W. Lin,Y. L. Wu, “A fast updated frequent pattern tree’’, presented at The 2006 IEEE International Conference on Systems, Man, and Cybernetics, 2006, Taiwan.

34. C. W. Lin, T. P. Hong, W. H. Lu,"The Pre-FUFP algorithm for incremental mining", Expert Systems with Applications 36 (5), 9498-9505 -2009

35. R. Amornchewin, W. Kreesuradej, “Incremental Association Rule Mining Using Promising Frequent Itemset Algorithm”, In: 6th International Conference on Information, Communications & Signal Processing, pp. 1–5 (2007)

36. C. H. Yang, D. L Yang, “IMBT-A Binary Tree for Efficient Support Counting of Incremental Data Mining”, CSE (1) 2009: 324-329

37. C. Bhadane, K. Shah, P. Vispute, “An Efficient Parallel Approach for Frequent Itemset Mining of Incremental Data”, International Journal of Scientific & Engineering Research 3(2) (February 2012)

38. C. F. Ahmed, S. K. Tanbeer, B. Jeong, "Single-pass incremental and interactive mining for weighted frequent patterns," Expert Systems with Applications, vol. 39, Issue 9, 2012, pp. 7976-7994.

39. M. Zongmin,"Intelligent Databases: Technologies and Applications", IGI publishing, 320 pages, 2007

40. C. C. Aggarwal, “Managing and Mining Uncertain Data”, Kluwer Press, 2009.

41. C. C. Aggarwal, Y. Li, J. Wang, and J. Wang, “Frequent pattern mining with uncertain data”, KDD, pp. 29–38, 2009.

42. C. C. Aggarwal, and P. S. Yu, “A survey of uncertain data algorithms and applications”, IEEE Transactions on Knowledge and Data Eng., 21(5): pp.609–623, 2009.

43. T. Bernecker, H.-P. Kriegel, M. Renz, F. Verhein, and A. Zufle, “Probabilistic frequent itemset mining in uncertain databases”, KDD, pp.119–128, 2009.

44. Y. Tong, L. Chen, Y. Cheng, and P. S. Yu, “Mining Frequent Itemsets over Uncertain Databases”, Proc. VLDB Conference, PVLDB, Vol. 5, 2012.

45. C. K. Chui, B. Kao, and E. Hung, "Mining frequent itemsets from uncertain data", The Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD), pp.47-58, 2007.

46. C. K.-S. Leung, M. A. F. Mateo, and D. A. Brajczuk, "A tree-based approach for frequent pattern mining from uncertain data", PAKDD, pp. 653-661, 2008.

47. Y. Liu, K. Liu, and M. Li, “Passive diagnosis for wireless sensor networks”, IEEE/ACM Trans. Netw.,18(4):1132–1144, 2010

48. S. Suthram, T. Shlomi, E. Ruppin, R. Sharan, and T. Ideker, "A direct comparison of protein interaction confidence assignment schemes", BMC Bioinformatics, 7:360, 2006.

49. C.C. Aggarwal, “On Unifying Privacy and Uncertain Data Models,” Proc. 24th IEEE Int’l Conf. Data Eng. (ICDE), 2008 A. Motro, P. Smets, “Uncertainty Management in Information Systems”, ISBN 978-1-4615-6245-0, 1997.

50. C. H. Cai, A. W. Chee Fu, C. H. Cheng, and W. W. Kwong. “Mining Association Rules with Weighted Items,” Proceedings of the Sixth International Conference on Intelligent Data Engineering and Automated Learning (IDEAL 2005), July 1998.

51. F. Tao, “Weighted Association Rule Mining Using Weighted Support and Significant Framework,” Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 661-666, Aug. 2003

52. W. Wang, J. Yang, and P. S. Yu, “Efficient Mining of Weighted Association Rules (WAR),” Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 270-274, Aug. 2000.

53. U. Yun, and J. J. Leggett, “WFIM: Weighted Frequent Itemset Mining with a Weight Range and a Minimum Weight,” Proceedings of the Fourth SIAM International Conference on Data Mining, pp. 636-640, April 2005.

54. U. Yun, “Efficient Mining of weighted interesting patterns with a strong weight and/or support affinity“. Information Sciences 177, 3477–3499 (2007).

55. M. Alharbi, S. Pathak, S. Rajasekaran,"Frequent Itemsets Mining on Weighted Uncertain Data", Signal Processing and Information Technology (ISSPIT), 2014 IEEE Symposium on 15-17 December 2014.

56. A. A. Nanavati, K. P. Chitrapura, S. Joshi, and R. Krishnapuram, "Mining generalized disjunctive association rules", CIKM, ACM, pp. 482-489, 2001.

57. M. C. Sampaio, Fernando H. B. Cardoso, Gilson P. dos Santos Jr.,Lile Hattori, "Mining Disjunctive Association Rules", 15 aug. 2008

58. M. Alharbi, P. Periaswamy, S. Rajasekaran,"Disjunctive rules mining from uncertain databases", Computers and Communication (ISCC), 2014 IEEE Symposium on 23-26 June 2014.

59. L. Jiuyong, L. ThucDuy, L. Lin, L. Jixue, J. Zhou,S. Bingyu, "Mining Causal Association Rules", 2013 IEEE 13th International Conference on Data Mining Workshops (ICDMW) - TX, USA - Dec. 7, 2013 to Dec. 10, 2013

60. J. Bowes, E. Neufeld, J. E. Greer,J. Cooke, "A Comparison of Association Rule Discovery and Bayesian Network Causal Inference Algorithms to Discover Relationships in Discrete Data", in Discrete Data, Proceedings of the Thirteenth Canadian Artificial Intelligence Conference -AI'2000

61. TC. Chalmers, H. Smith, B. Blackburn, B. Silverman, B. Schroeder, D. Reitman, A. Ambroz , "A method for assessing the quality of a randomized control trial". Controlled Clinical Trials 1981.

62. L. Jiuyong, L. Lin, L. ThucDuy, "Practical Approaches to Causal Relationship Exploration", Springer International Publishing, 2015, SpringerBriefs in Electrical and Computer Engineering

63. I. Good, “A theory of causality,” British Journal for the Philosophy of Science 9, pp. 307–310, 1959.

64. H. Reichenbach, “The principle of causality and the possibility of its empirical confirmation,” Routledge and Kegan Paul, London, 1923.

65. P. Suppes, “A probabilistic theory of causality,” North-Holland, Amsterdam, 1970.

66. G. Grahne, L. Lakshmanan, and X. Wang, "Efficient mining of constrained correlated sets," In Proc. 2000 Int. Conf. Data Engineering (ICDE-00), San Diego, CA, pp. 512-521, 2000.

67. J. Pearl, and T. S. Verma, “A theory of inferred causation,” in Knowledge Representation and Reasoning: Proceedings of the Second International Conference, 1991, pp. 441–452.

68. D. Heckerman, “A Bayesian approach to learning causal networks,” in UAI, 1995, pp. 285–295.

69. D. Heckerman, “Bayesian networks for data mining”, Data Mining and Knowledge Discovery 1, pp. 79–119, 1997.

70. S. Nadkarni and P. P. Shenoy, “A Bayesian network approach to making inferences in causal maps,” European Journal of Operational Research 128(3), pp. 479–498, 2001.

71. J. Pearl, “From Bayesian network to causal networks,” in Bayesian Networks and Probabilistic Reasoning, pp. 1–31, 1994.

72. M. R. Waldmann and L. Martignon, “A Bayesian network model of causal learning,” in Proceedings of the Twentieth Annual Conference of the Cognitive Science Society, 1998, pp. 1102–1107.

73. J. Pearl, "Causality: Models, Reasoning, and Inference," Cambridge University Press, 2000.

74. D. M. Chickering, D. Heckerman, and C. Meek, “Large-sample learning of Bayesian networks is NP-hard”, JMLR 5, 1287–1330, 2004.

75. C. F. Aliferis, A. Statnikov, I. Tsamardinos, S. Mani, X. D. Koutsoukos, “Local causal and Markov blanket induction for causal discovery and feature selection for classification Part I: Algorithms and empirical evaluation,” JMLR 11,pp. 171–234, 2010.

76. S. Mani, G. F. Cooper, P. Spirtes, “A theoretical study of y structures for causal discovery”, in UAI, 2006, AUAI Press

77. J. P. Pellet, “Using Markov blankets for causal structure learning,” JMLR 9, pp. 1295–1342, 2008.

78. C. Silverstein, S. Brin, R. Motwani, J. Ullman, “Scalable techniques for mining causal structures,” Data Mining and Knowledge Discovery 4, pp. 163–192, 2000.

79. G. F. Cooper, “A simple constraint-based algorithm for efficiently mining observational databases for causal relationships,” Data Mining and Knowledge Discovery 1, pp. 203–224, 1997.

80. N. Mantel and W. Haenszel, “Statistical aspects of the analysis of data from the retrospective analysis of disease”, Journal of the National Cancer Institute, Vol. 22, No. 4, pp. 719-748,1959.

81. M. W. Birch, “The Detection of Partial Association, I: The 2×2 Case”, Journal of the Royal Statistical Society, Vol. 26, No. 2, pp. 313-324, 1964.

82. J. Zhou , L. Jiuyong, L. Lin, L. Jixue, L. ThucDuy, S. Bingyu, W. Rujing, " Discovery of Causal Rules Using Partial Association ", Data Mining (ICDM), 2012 IEEE 12th International Conference on 10-13 Dec. 2012.

83. S. Brin, R. Motwani, C. Silverstein, “Beyond market baskets: Generalizing association rules to correlations,” In: Proceedings ACM SIGMOD International Conference on Management of Data, Tucson, Arizona, USA, May 13-15, pp.265–276. ACM, New York – 1997

84. http://nugget.unisa.edu.au/Causalbook/

85. http://www.phil.cmu.edu/tetrad/.

86. M. Kalisch P. B¨uehlmann and M. H. Maathuis. Variable selection for high-dimensional linear models: partially faithful distributions and the PC-simple algorithm. Biometrika, 7:261–278,2010.

87. C. F. Aliferis, A. Statnikov, I. Tsamardinos, S. Mani, and X. D. Koutsoukos. Local causal and Markov blanket induction for causal discovery and feature selection for classification Part I: Algorithms and empirical evaluation. Journal of Machine Learning Research, 11:171–234,2010.

88. M. Alharbi, S. Rajasekaran," Conjunctive Combined Causal Rules Mining", Signal Processing and Information Technology (ISSPIT), 2014 IEEE Symposium on 7-10 December 2015.

89. M. Alharbi, S. Rajasekaran," Disjunctive Combined Causal Rules Mining", Signal Processing and Information Technology (ISSPIT), 2014 IEEE Symposium on 7-10 December 2015.