

Algorithms for Mining Association Rules from Large Transactional Distributed Data

Submitted in partial fulfilment for the award of the degree of

DOCTOR OF PHILOSOPHY

In

Department of Computer Science & Engineering

By

MANOJ SETHI

Roll No. 2k12/PHDCO/05

Under the Supervision of

Prof. (Mrs.) Rajni Jindal

Professor & Head of Department

Department of Computer Science & Engineering,
Delhi Technological University, Delhi – 110042, India



Delhi Technological University,
Shahbad Daulatpur, Main Bawana Road,
Delhi – 110042, India

2021

CERTIFICATE



DELHI TECHNOLOGICAL UNIVERSITY

(Govt. of National Capital Territory of Delhi)

BAWANA ROAD, DELHI – 110042

Date: _____

This is to certify that the work embodied in the thesis titled “Algorithms for Mining Association Rules from Large Transactional Distributed Data” has been completed by Manoj Sethi under my supervision and guidance towards fulfilment of the requirements for the degree of Doctor of Philosophy of Delhi Technological University, Delhi. This work is based on original research and has not been submitted in full or in part for any other diploma or degree of any university.

Prof. (Mrs.) Rajni Jindal

Professor & Head,

Department of Computer Science & Engineering,

Delhi Technological University

Abstract

Lots of advancements in the database technologies in the last few decades attracted researchers to work in this area. Databases which were mostly centralised have been changed to distributed databases where data is partitioned and stored at different locations, because of the availability of modern technologies, fast network, internet, increased size of data and industry demand. Centralised database are also used for creating data warehouses and then data mining for getting some useful information for the critical decisions in the area of education, medical, commercial and many more. Now, with the increasing demand of the distributed databases, mining data ware-house concept is also changing to distributed data mining where mining is done on the data partitions stored at different locations and then the aggregation or merging the results is done for the global mining.

Distributed data mining (DDM) has become important research area with the increase in large distributed transactional databases and we need to investigate important patterns in such databases. On one side distributed processing may increase not only the processing capabilities but also increases the cost of communication and storage cost. The work focuses on the distributed data association rules mining for the transactional data. It has opened new areas of research to develop the architecture, framework and algorithms in the area of distributed data mining. The distributed data partitions where data is created or generated at different locations vary in size, and number of frequent patterns are generated at different locations. This area is not very old and little work has been done in the distributed data association rules mining. Some new algorithms and new data structures are proposed in literature. Algorithms which are available, mostly first partition the database, distribute them amongst different sites for parallel processing. In the real life

scenario data generated at different sites is not under the control of centralised database and the numbers of transactions at each site are highly varied. Due to this, some sites are heavily loaded and some sites are comparatively free, research is to be focused on these issues. Distributed mining is used in many commercial areas and there is a need to explore new commercial applications of the data mining.

This work focuses on the study of the recent development in this area of distributed data association rule mining (DARM). It highlights the issues and challenges, their correlation, available technologies and tools, different algorithms, real data repositories for mining in the area of DARM. On the basis of the study, work focuses on the development of new algorithms and developing new application model addressing different issues and challenges in the area of DARM. Datasets Mushroom, Connect, T10I40D400K and Chess from the fimi data repository are taken for the implementation and testing of the proposed algorithms. Application model is developed and implemented on the actual data of a tour & travel company for the last 2 years.

A new algorithm named as QDFIN(Quick distributed frequent itemset mining using nodeset) is proposed in this research which uses the efficient nodeset data structure to store the candidate itemsets locally at each site and zero-first technique to balance the load and pruning to reduce the candidate sets. The algorithm is implemented and the speed performance is compared with some of the existing algorithms FDM and PFIN on Mushroom dataset. Results shows that the proposed algorithm not only outperforms other algorithms on varying size data partition but also, on uniform distributed data on 4, 5 and 6 nodes setups.

A novel approach, size based assignment, is proposed in this work which takes care of the database size available at each site while distributing the load for finding the

global frequent itemsets. It also reduces the communication load by pruning and no-broadcasting techniques. The algorithm is compared with FDM and PFIN on execution time on mushroom, connect, chess and T10I4D100K datasets. Results show that the new technique performed best amongst them in time execution and is best load balancing technique.

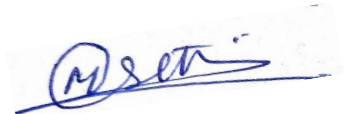
The application area chosen for the study is a tour and travel company organizing package tours, because tourism industry is growing very fast and small, medium and large sized companies are operating in this area. Tourism is a potential application where mining can be applied and new association rules can be generated which can help the companies to develop new strategies and target potential customer based on the mining outcome. This work applies the distributed data mining technique on a medium sized tour & travel company for finding the association between age and destination visited parameters. The results show that association rules generated by mining are useful and effective for the growth of the business and making new strategies.

DECLARATION

I, Manoj Sethi, Ph.D. student (Roll No. 2k12/PHDCO/05), hereby declare that the thesis entitled “**Algorithms for Mining Association Rules from Large Transactional Distributed Data**” which is submitted for the award of the degree of Doctor of Philosophy in Computer Science & Engineering, is a record of research work carried out by me in the Department of Computer Science & Engineering, Delhi Technological University. I further declare that this work is based on original research and has not been submitted to any university or institution for any degree or diploma.

Date:

Place: New Delhi



(Manoj Sethi)

2k12/PHDCO/05

Department of Computer Science & Engineering

Delhi Technological University (DTU)

New Delhi -110042

ACKNOWLEDGMENT

First and foremost I am extremely grateful to my supervisor, Prof. Rajni Jindal for her invaluable advice, continuous support, and patience during my PhD study. Her immense knowledge and plentiful experience have encouraged me in all the time of my academic research and daily life. She was always motivating me for the study and giving useful inputs time to time.

I would also like to thank my colleague friends in the department for their direct and indirect technical support on my study.

Finally, I would like to express my gratitude to my parents, wife and children. Without their tremendous understanding and encouragement in the past few years, it would be impossible for me to complete my study.

Manoj Sethi

Table of Contents

CERTIFICATE	i
ABSTRACT	ii
DECLARATION	v
ACKNOWLEDGEMENT	vi
TABLE OF CONTENTS	vii
LIST OF TABLES	x
LIST OF FIGURES	xiii
LIST OF ABBREVIATIONS	xv
1 INTRODUCTION.....	1
1.1 DATA MINING.....	1
1.1.1 Data Mining Techniques	1
1.2 ASSOCIATION RULE MINING	2
1.2.1 Distributed Data Mining	4
1.3 MOTIVATION	6
1.4 PROBLEM STATEMENT AND OBJECTIVES.....	7
1.5 RESEARCH CONTRIBUTION.....	8
1.6 ORGANIZATION OF THESIS.....	9
2 LITERATURE SURVEY	11
2.1 Distributed ASSOCIATION RULE MINING	11
2.1.1 Basic Terms of Frequent Pattern Mining Algorithm	16
2.2 DATA SOURCES	17
2.3 DATA MINING TOOLS AND SOFTWARE	18
2.4 ASSOCIATION RULE MINING ALGORITHMS	19
2.4.1 Data Structures Used in Data Mining	25
2.4.2 Issues And Challenges.....	26
2.5 APPLICATION MODELS OF DARM	28
2.6 RESEARCH GAP	31
3 METHODOLOGY	33
3.1 DISTRIBUTED DATA MINING PROCESS	33
3.1.1 Assumptions	33
3.1.2 Process Activities and Definition	34
3.2 DISTRIBUTED DATA MINING MODEL.....	36
3.3 CANDIDATE SET REDUCTION BY PRUNING	37
3.4 NO BROADCASTING OF FREQUENT ITEMSETS	38

3.5	NODESET DATA STRUCTURE.....	39
3.5.1	POC Tree.....	39
3.5.2	Nodesets.....	40
3.6	ZERO-FIRST TECHNIQUE.....	41
3.7	SIZE BASED ASSIGNMENT TECHNIQUE.....	43
3.8	COMPARISON METRICS.....	45
3.9	SUMMARY.....	47
4	<i>QUICK DISTRIBUTED FREQUENT ITEMSET MINING USING NODESET.....</i>	48
4.1	PROPOSED ALGORITHMS.....	48
4.1.1	Efficiency of Local Frequent Mining.....	54
4.1.2	Distributed Database and Resources.....	54
4.1.3	Communication Load Reduction.....	55
4.2	EXPERIMENTAL EVALUATION.....	55
4.2.1	Experimental setup.....	56
4.3	PERFORMANCE ANALYSIS.....	57
4.3.1	Comparison on Uniform Data Partition Size.....	57
4.3.2	Comparison on Varying Data Partition Size.....	60
4.4	SUMMARY.....	64
5	<i>SIZE BASED DISTRIBUTED ASSOCIATION RULE MINING.....</i>	65
5.1	PROPOSED ALGORITHMS.....	65
5.1.1	Efficiency at Each Site.....	68
5.1.2	Low Communication Overhead.....	69
5.2	EXPERIMENTAL EVALUATION.....	70
5.2.1	Experimental setup.....	70
5.3	PERFORMANCE ANALYSIS.....	71
5.3.1	Generating Local Itemsets.....	72
5.3.2	Polling sites assignment.....	74
5.3.3	Discussion.....	76
5.4	COMPARISON OF SBDARM AND QDFIN.....	79
5.4.1	Discussion.....	80
5.5	SUMMARY.....	81
6	<i>APPLICATION MODEL – A CASE STUDY.....</i>	82
6.1	METHODOLOGY.....	82
6.1.1	Distributed Mining of Association Rules.....	82
6.1.2	Assumptions.....	83
6.1.3	Algorithm.....	83
6.2	IMPLEMENTATION AND RESULTS ANALYSIS.....	84
6.2.1	Experiment 1: Minimum support threshold = 0.200.....	89
6.2.2	Experiment 2: Minimum support threshold = 0.100.....	91
6.2.3	Relationships Between Local and Global Rules.....	93
6.3	SUMMARY.....	95
7	<i>CONCLUSIONS.....</i>	96

7.1	FUTURE WORK.....	98
	<i>PAPERS PUBLISHED</i>	99
	<i>REFERENCES</i>	100

List of Tables

TABLE 2.1	DATA MINING TOOLS & SOFTWARE	18
TABLE 2.2	INVERSELY PROPORTIONAL ISSUES	28
TABLE 2.3	DIRECTLY PROPORTIONAL ISSUES	28
TABLE 3.1	SAMPLE DATABASE TRANSACTIONS	39
TABLE 3.2	DATASETS AND THEIR SPECIFICATIONS	46
TABLE 4.1	SPECIFICATIONS OF MUSHROOM DATASET	56
TABLE 4.2	UNIFORM DATA PARTITION SIZES AT DIFFERENT SITES	56
TABLE 4.3	VARYING DATA PARTITION SIZES AT DIFFERENT SITES	56
TABLE 4.4	EXECUTION TIME ON UNIFORM PARTITION SIZE ON 4 NODES (SECONDS)	58
TABLE 4.5	EXECUTION TIME ON UNIFORM PARTITION SIZE ON 5 NODES (SECONDS)	58
TABLE 4.6	EXECUTION TIME ON UNIFORM PARTITION SIZE ON 6 NODES (SECONDS)	59
TABLE 4.7	EXECUTION TIME ON VARYING PARTITION SIZE ON 4 NODES (SECONDS)	61
TABLE 4.8	EXECUTION TIME ON VARYING PARTITION SIZE ON 5 NODES (SECONDS)	61
TABLE 4.9	EXECUTION TIME ON VARYING PARTITION SIZE ON 6 NODES (SECONDS)	63
TABLE 5.1	SPECIFICATIONS OF THE DATASETS USED	70
TABLE 5.2	DETAILS OF THE DATA PARTITIONS AVAILABLE AT DIFFERENT SITES FOR DIFFERENT DATASETS	71
TABLE 5.3	LOCAL FREQUENT 1-ITEMSETS GENERATED AT EACH PARTITION ON MUSHROOM DATASET	72

TABLE 5.4	LOCAL FREQUENT 1-ITEMSETS GENERATED AT EACH PARTITION ON CONNECT DATASET	73
TABLE 5.5	LOCAL FREQUENT 1-ITEMSETS GENERATED AT EACH PARTITION ON CHESS DATASET	73
TABLE 5.6	LOCAL FREQUENT 1-ITEMSETS GENERATED AT EACH PARTITION ON T10I4D100K DATASET	73
TABLE 5.7	PRUNING SITES ASSIGNMENT BY SBDARM TO LOCAL FREQUENT 1-ITEMSETS ON MUSHROOM DATASET	74
TABLE 5.8	PRUNING SITES ASSIGNMENT BY SBDARM TO LOCAL FREQUENT 1-ITEMSETS ON CONNECT DATASET	75
TABLE 5.9	PRUNING SITES ASSIGNMENT BY SBDARM TO LOCAL FREQUENT 1-ITEMSETS ON CHESS DATASET	75
TABLE 5.10	PRUNING SITES ASSIGNMENT BY SBDARM TO LOCAL FREQUENT 1-ITEMSETS ON T10I4D100K DATASET	76
TABLE 6.1	TOUR AND TRAVEL COMPANIES' CLASSIFICATION	84
TABLE 6.2	DATA TABLES USED - BOOKING-MASTER AND TOURISTS-DETAILS	85
TABLE 6.3	DATA ATTRIBUTES TAKEN FOR ANALYSIS	85
TABLE 6.4 (A)	SAMPLE DATA TABLE	86
TABLE 6.4 (B)	SAMPLE DATA TABLE AFTER TRANSFORMATION	86
TABLE 6.5	DATASETS SIZES AVAILABLE AT VARIOUS SITES	87
TABLE 6.6	LOCAL SUPPORT COUNT (%) AT EACH SITE	88
TABLE 6.7	SUPPORT AND CONFIDENCE THRESHOLD VALUES	89
TABLE 6.8	CANDIDATE SETS FOR SUPPORT COUNT THRESHOLD = 0.200	89
TABLE 6.9	GLOBAL SUPPORT AND CONFIDENCE	89
TABLE 6.10	ASSOCIATION RULES FOR SUPPORT THRESHOLD 0.200 AND CONFIDENCE THRESHOLD 0.500	90
TABLE 6.11	ASSOCIATION RULES FOR SUPPORT THRESHOLD 0.200 AND CONFIDENCE THRESHOLD 0.300	90

TABLE 6.12	CANDIDATE SETS FOR SUPPORT THRESHOLD = 0.100	91
TABLE 6.13	GLOBAL SUPPORT AND CONFIDENCE	91
TABLE 6.14	ASSOCIATION RULES FOR SUPPORT THRESHOLD 0.100 AND CONFIDENCE THRESHOLD 0.500	92
TABLE 6.15	ASSOCIATION RULES FOR SUPPORT THRESHOLD 0.100 AND CONFIDENCE THRESHOLD 0.300	92

List of Figures

FIGURE 1.1	CENTRALIZED DATA MINING	3
FIGURE 1.2	DISTRIBUTED DATA MINING	5
FIGURE 3.1	A GENERAL PURPOSE DARM ARCHITECTURE	37
FIGURE 3.2	PROPOSED MODEL FOR DARM	37
FIGURE 3.3	POC TREE CONSTRUCTION	40
FIGURE 4.1	EXECUTION TIME ON UNIFORM PARTITION SIZE ON 4 NODES	58
FIGURE 4.2	EXECUTION TIME ON UNIFORM PARTITION SIZE ON 5 NODES	59
FIGURE 4.3	EXECUTION TIME ON UNIFORM PARTITION SIZE ON 6 NODES	60
FIGURE 4.4	EXECUTION TIME ON VARYING PARTITION SIZE ON 4 NODES	61
FIGURE 4.5	EXECUTION TIME ON VARYING PARTITION SIZE ON 5 NODES	62
FIGURE 4.6	EXECUTION TIME ON VARYING PARTITION SIZE ON 6 NODES	63
FIGURE 5.1	EXECUTION TIME ON MUSHROOM DATASET	77
FIGURE 5.2	EXECUTION TIME ON CONNECT DATASET	77
FIGURE 5.3	EXECUTION TIME ON CHESS DATASET	78
FIGURE 5.4	EXECUTION TIME ON T10I4D100K DATASET	78
FIGURE 5.5	COMPARISON OF SBDARM AND QDFIN ON CHESS DATASET	80
FIGURE 5.6	COMPARISON OF SBDARM AND QDFIN ON T10I40D400K DATASET	80
FIGURE 6.1	AGE GROUP AND THEIR SUPPORT COUNT	87
FIGURE 6.2	DESTINATIONS FREQUENCY -SUPPORT COUNT	88
FIGURE 6.3	LOCAL SUPPORT COUNT AT VARIOUS SITES AND GLOBAL SUPPORT	93

FIGURE 6.4	LOCAL SUPPORT COUNT AT VARIOUS SITES AND GLOBAL CONFIDENCE	94
FIGURE 6.5	GLOBAL SUPPORT COUNT AND GLOBAL CONFIDENCE	94

List of Abbreviations

ARM	Association Rule Mining
Can.D	Candidate Distribution
CD	Count Distribution
CPU	Central Processing Unit
CREDLM	Chain Retail Enterprise based on Dispersed Learning
CREMLM	Chain Retail Enterprise based on Massed Learning
DARM	Distributed Association Rule Mining
DD	Data Distribution
DDM	Distributed Decision Miner
DDM	Distributed Decision Mining
DMA	Distributed Mining of Association Rules
DMCA	Distributed Mining Association Rules with Item Constraints
DMCI	Distributed Mine Closed Itemsets
DP3	Distributed PrePostPlus
DPO	Frequent Patterns Organization
DRHPDM	Data source Relevance-based Hierarchical Parallel Distributed Data Mining
DT-DPM	Decomposition Transaction for Distributed Pattern Mining
e-CRM	electronic Customer Relationship Management
Eclat	Equivalence Class Clustering and bottom-up Lattice Traversal
FCET	Frequent Closed Enumeration Table
FCI	Frequent Closed Itemset
FDM	Fast Distributed Mining of association rules
FI	Frequent Itemset

FIMI	Frequent Itemset Mining Implementation
FIN	Frequent Itemset Mining
FP-Tree	Frequent-Pattern Tree
FPO	Frequent Patterns Organization
GIS	Geographical Information System
MDFI	Multi-Relation Data Mining
MR-CIRD	Map-Reduce based Consistent and Inconsistent Rule Detection
MRDM	Multi-Relation Data Mining
ODAM	Optimized distributed association rule mining
PARM	Parallel Association Rule Mining
PDM	Parallel Data Mining
PFIN	Parallel Frequent Itemset Mining
PPF	Parallel FP-Growth
POC-Tree	pre-order Coding Tree
PPC-Tree	Pre-order Post-order Code Trees
QDFIN	Quick Frequent Itemset Mining using Nodeset
R-Apriori	Reduced-Apriori
SARM	Sequential Association Rule Mining
SBDARM	Size Based Distributed Association Rule Mining

1 INTRODUCTION

1.1 DATA MINING

Data mining is a process to discover unknown valid patterns and relationships in large data. Process of analyzing data, finding correlations or patterns and summarizing them are known as Data mining. It is also referred to as knowledge discovery. Data Mining involves analyzing and extracting useful information from data. There are many applications of data mining such as Fraud Detection, Market Analysis, Biological Analysis, Science Exploration etc.

1.1.1 Data Mining Techniques

- Association is mining relationships or associations amongst data. It is to find interesting and hidden frequent patterns or items and to determine association rules.
- Classification describes the data concepts, groups or classes to predict the class of objects for which the class is not known and to judge about the type of customer, item, or object. It is used to identify a particular class by describing multiple attributes.
- Clusters - Similar objects in the data are grouped into clusters based on data similarity like logical relationships or consumer preferences to identify market segments or consumer affinities.

- Prediction means to predict the missing or unavailable data values. Generally, regression analysis technique is used for prediction.
- Outlier Analysis is to identify the data objects that do not comply with general behaviors of the data available, and it may be used for fraud deduction in credit cards.
- Evolution Analysis is description and model regularities or trends for objects whose behavior changes over time.
- Mining Stream is extracting knowledge from the continuous and rapid data records, Time-Series and Sequence Data.
- Graph Mining - Analysis of real graphs, effect on applications, model to generate realistic graphs.
- Social Media mining - Applying data mining methods to huge data sets can improve search results for everyday search engines, realize specified target marketing for business, help psychologists study behavior etc.
- Multi relational Data mining provides the mining in multiple tables directly. In MRDM the patterns are available in multiple tables (relations) from a relational database.

1.2 ASSOCIATION RULE MINING

Association rule mining means finding interesting associations, correlations and frequent patterns amongst a large number of items or objects that are contained in a transaction database, relational database or some other kind of data repository. It helps in decision making process for business purposes like in supermarkets for catalogue design,

basket data analysis, planning etc. Various algorithms have been proposed to find the frequent itemsets. Association rule mining was first proposed by R. Agrawal [60]. After that, a lot of research has been done in this area and new approaches and algorithms have been proposed. Association rules mining has many applications and it is now widely used in sales, tourism, medical, marketing etc. Globalization has opened new challenges and opportunities for the researchers in association rule mining due to the increased volume of data and changed characteristics of data.

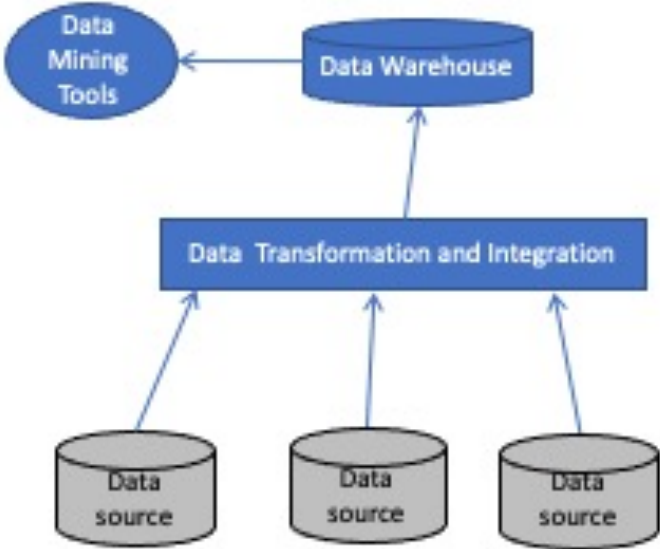


Figure 1.1 Centralized data Mining

Centralized database environment is created at one centralized location with all the resources and nodes accessing the database are connected to this database through communication links which is shown in Figure 1.1. Due to the Fast growth and use of the information technology in different domain and globalization, not only large amount of data is being created but the need for readily available information also increases. It

requires more and more processing capabilities, storage and data communication and other resources. Doing mining tasks at one location has become inefficient. The use of distributed database environment in place of centralized databases is increased, in which database is portioned and stored at different locations to share the resources and perform parallel processing.

Apriori Algorithm is one of the most popular sequential mining algorithms for mining frequent itemsets. Apriori uses prior knowledge of frequent itemset properties. It employs an iterative level-wise approach where $(k-1)$ -itemsets are used to explore k -itemsets. Each frequent k -itemset finding requires a full database scan.

1.2.1 Distributed Data Mining

The voluminous amount of data is created at different sites and locations which varies in size. It is not feasible to transfer all data onto a centralized database for analysis due to the limitation of resource, communication links or sometime due to the policies or privacy issues. With the rising size of data and the demand of mining patterns from data, there is a need to find a solution to analyse the data as and where it is generated or gathered or created and find interesting and useful frequent patterns in the distributed data.

The solution is Distributed Association Rule Mining (DARM) [23]. In DARM, data is stored at different locations or sites and various processors work parallel to provide fast and efficient results. Distributed data mining finds local frequent patterns at different sites, communicates with other sites and finds the global frequent itemsets. Figure 1.2 shows a general distributed mining architecture. As compared to centralized frequent itemsets mining less number of algorithms have been proposed in literature for DARM.

Several algorithms are available for association rules mining the centralized database. These algorithms available for the centralized database mining cannot be used directly for DARM. DARM requires local processing at all the sites to find the local frequent itemsets, communicate between the sites and finally find frequent global itemsets. Parallel frequent mining algorithms are also not fit for that, rather algorithms which consider the number of transactions available at each site and distribute the workload to all sites are required for optimum use of the computational capabilities available at each site. Some DRAM [23] algorithms are proposed in the literature.

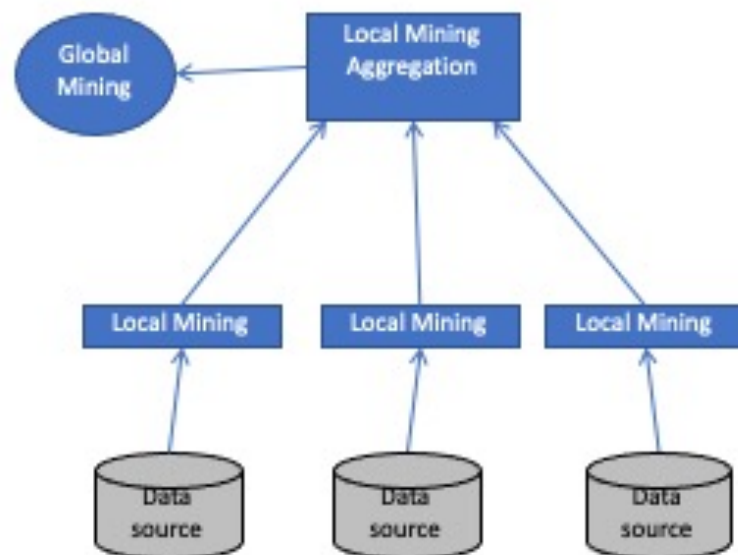


Figure 1.2 Distributed data mining

Steps which are generally involved in the DARM are scanning databases for finding local frequent itemsets, storage of frequent itemsets, local pruning to reduce the candidate sets, sharing local counts amongst different sites, global pruning to reduce the itemsets, finding Association rule etc. This requires huge storage space, communication, synchronization and processing capabilities and trade-offs between them.

1.3 MOTIVATION

With the massive growth of real time business applications, online transaction analysis is very much essential. As the number of transactions increase, it becomes very difficult to determine the frequent itemsets with less time and space complexities. Due to the high volume and limitation of resources at centralized site, database is not centralized but distributed in nature and kept as and where it is created or gathered. With day-to-day operations, number of transactions at different sites may vary from a few hundred to lacs of transactions which creates imbalance in the data sizes available at different sites and also processing load on these sites. It may not be feasible to transfer data from all sites to one centralised site for processing. Also, large database mining requires substantial processing power and distributed mining processing at each site enhances the processing capabilities of the system by dividing the load. In parallel and distributed frequent pattern mining algorithms, each site is responsible to extract the knowledge by exchanging or broadcasting the data required by other sites. Several algorithms have been proposed for association rule mining, but they are ineffective with very large distributed imbalanced datasets.

- There is a need to develop algorithms which focus on the local performance at each site by reducing I/O operations and managing data effectively.
- Algorithms are required that utilise the site resources in a balanced manner by allocating load based on the local load.
- Reducing communication load can improve the overall performance of the algorithms, is another factor that is to be addressed.

- Association mining is important tool for business growth, particularly in the area of market basket analysis, which is commonly used in the business. So, exploiting the areas of applications of the distributed data mining for different industries and widen its use is also required.

1.4 PROBLEM STATEMENT AND OBJECTIVES

The association rule mining on the distributed data has gained popularity because of the change in the nature of data which is more and more distributed across the globe. Most of the algorithms available for the association rule mining consider that the data available at each site is uniformly distributed. The distributed data environment where data is created or gathered, stored at different locations and need to be used without transfer to a single centralized site. Most of the available algorithms are not suitable for the setup where data size vary and efficient techniques are needed to take care of the non-uniform nature of data.

There is a need to develop methodology and DARM algorithms for finding association rules in the distributed imbalanced transactional data which can improve the performance by not only reducing the communication overhead, data scans and storage needs but also assign load to less occupied sites in order to get the best utilization of the resources.

Hence the problem statement of the research can be stated as:

“To build constructive and efficient distributed association rule mining algorithms for the varying distributed data partition sizes available at different sites.”

The problem has been evaluated with the following research objectives:

- Do a systematic literature review including study the existing Distributed Data

Mining (DDM) algorithms, issues and challenges for association rule mining of the distributed data.

- Discuss methodology for the design of distributed data mining tasks and applications on distributed data environments addressing the issues.
- Develop new and effective DDM algorithms for mining association rules with reduced cost and improved performance.
- Implement and compare proposed algorithms with the existing comparative algorithms to evaluate the performance.
- Develop an application model by conducting a business case study, implement on a real commercial transactional data and make suitable recommendations for the improvement of the business.

The research focus is on developing efficient algorithms and conducting a case study to develop an application model for a business house and recommend a solution.

1.5 RESEARCH CONTRIBUTION

The aim of this thesis was to make a contribution to the field of association rule mining on the distributed transactional data where the database is partitioned and stored at different sites where the actual data is created. The study has evolved the methodology for the development of efficient algorithms for association rule mining for the distributed database, evaluation and comparison of the same with the existing ones for making interesting rules for the business houses.

The primary contributions of this research are:

- Comprehensive review and survey of current association rule mining techniques for centralized as well as distributed data environments and the process involved in the distributed association rule mining.
- Study of the issues and challenges in the development of the algorithms of DARM and the effect of the issues on the performance of the algorithms. It also highlighted the relationships amongst the issues which are directly proportional or inversely proportional issues. Addressing one issue may adversely affect the other issue and there is a need to have a trade-off between them.
- Developed techniques to address different issues, and to handle the distributed data nature i.e., varying data partitions size in terms of number of transactions at each site.
- Algorithms are proposed based on the techniques developed to improve the performance of association rules mining for the distributed data.
- Experimental evaluations are conducted on the different real and synthetic datasets and the results prove that proposed works achieve their aims. The proposed algorithms improve the time performance for different setups and different datasets by effectively reducing the ideal time of the resources at different sites, reducing the communication load and data scans.
- Theoretical knowledge and formulas are applied to a real-world problem and an application model for a business house, as a case study, is developed and important useful association rules are recommended.

1.6 ORGANIZATION OF THESIS

The rest of the dissertation report is organized as follow:

Chapter 2 discusses the review of the literature on the frequent pattern mining algorithms including the algorithms of distributed data mining based on the research contributions that have been done, up to the date of this research. It shows the different issues and challenges and their relation in developing the distributed data mining algorithms. It examines the activities involved in any data mining process. Finally, it explores the availability of the different data repositories for the mining research.

Chapter 3 discusses the available and proposed methodologies for addressing different challenges and issues involved in the development of the proposed algorithms, illustrates them with the help of suitable examples.

Chapter 4 explains the proposed algorithm “Quick Distributed Frequent Itemset Mining using Nodaset” (QDFIN) in detail. It explains the algorithm, its implementation and comparison with the existing algorithms. Detail discussion on the results included in the chapter.

Chapter 5 discusses in detail, the steps of the second proposed algorithm “Size Based Distributed Association Rule Mining” (SBDARM), its evaluation and comparison based on the experiments results. It also compares and discusses both the proposed algorithms.

Chapter 6 presents an application model of DARM based on the case study of a medium sized tour and travel company. It implements the model and finds the interesting rules for the company for the development and enhancement of the business.

Chapter 7 concludes the work with recommendations and also presents the future scope of the work.

2 LITERATURE SURVEY

A lot of research papers have been published in the area of DDM. In the Literature review on DDM for finding association rules, more than 80 research papers are reviewed, algorithms, their variant and issues are studied. In the literature survey the algorithms for the distributed data mining for mining association rule were studied. As a next step in this direction, an extensive literature survey in this area was conducted. The survey includes detailed study of the algorithms, tools, datasets, issues and challenges, research gap, in developing the algorithms and applications in this area. Many surveys have been conducted in finding association rules in the database [33].

2.1 DISTRIBUTED ASSOCIATION RULE MINING

Association rule mining earned popularity with a publication of Apriori algorithm by Agrawal in 1993 [62] which has been cited more than 22000 times according to google scholar. The Apriori based algorithms are of “anti-monotone property” [4]. This is widely used in finding the association rules. These algorithms use an approach to create and test candidate sets [60]. AprioriTID and AprioriHybrid [60] are two variations of apriori in literature. AprioriTID [60] uses the database only once for finding the frequency of the items. AprioriHybrid [60] uses both, Apriori initially and AprioriTID at the end. [34] proposed FP-Growth algorithm based on a tree structure which is created after database scan for mining frequent itemsets, which is used to store the frequent itemsets and this algorithm is efficient than apriori algorithm. Algorithm PPC-Tree[87] is proposed in 2012 which uses a new data structure N-List based on PrePost. N-List is a novel vertical data

representation structure and it is originated from a FP-Tree like structure.

Algorithm FIN, Fast mining frequent itemset using Nodeset[85] structure, based on the PrePost algorithm is another algorithm proposed in 2014. Nodeset is created using POC-Tree but to store information of each node, it makes Nodeset structure using postorder or preorder of the node. [57] states that FIN algorithm is the most recent algorithm and fast in generating frequent itemsets. DFIN [84] algorithm uses diffnodesets structure which is also based on structure nodeset. In 2015 [86] proposed an algorithm based on PPC-Tree structure PrePost⁺. To discover the frequent itemsets, it sets enumeration search tree using the N-List structure. Structure linear prefix tree[59] composed of array forms was introduced, which minimizes the pointers between the nodes. IFIN⁺ algorithm[74] uses multi-physical computational units whereas [73] proposed a shared memory parallelism for improving single machine performance for the frequent itemset mining. Nadar proposed a new data structure nagNodeset[53] used in the algorithm nagFIN. The algorithm is based on the nodes in the prefix tree. [36] proposed an effective algorithm based on optimized matrix computation for multi-party data computation having different challenges.

A parallel algorithm that runs on a distributed database with uniform capabilities at different sites requires to divide the workload equally for best load [72]. In E-business lots of heterogeneous data is generated and are distributed [42]. Distributed data may be incompatible [77] and DDM based on the extension set can solve this problem. An optimum solution is obtained [58] from DDM as it is done in parallel. The data streams mining [28] is more challenging in the distributed environment. The distributed mining has advantage in terms of degree of parallelism and scalability. The issues are load balancing, minimizing communication cost and overlapping communication and

computation. Performance of any algorithm [9] is also effected by the number of nodes in the distributed data system. With the increase of the transactions or nodes, the performance improves. Execution time increases when the number of nodes increases but number of transactions reduces [31].

DDM systems can be classified [13] based on (i) parallel data mining agents to enhance working efficiency (ii) meta learning to improve the quality of selection of data mining algorithm and (iii) grid to mine in geographically distributed environment. An efficient data structure [20] is required to store the information in mining tasks based on the algorithms. Count Distribution (CD) [34] is a simple algorithm where data is parallelized and apriori algorithm runs parallel where local support count is found for each itemset and then communicated to each other site and hence the global frequent itemsets are found by all the sites. AprTidRec, an algorithm based on apriori algorithm was proposed in 2011 [4]. It is different from Apriori because it deploys only the joint step but no pruning step. It creates a record structure called tidRec and has lesser execution time than apriori algorithm. FDM [23] finds the locally large itemsets, which are sent to the assigned polling sites and then the global counts are found to finally find global frequent itemsets. The total support count message exchange is just $O(n)$. It mainly uses apriori and CD algorithms.

DMA [24] is another algorithm for distributed association rule mining, which generates a small candidate sets and $O(n)$ messages are exchanged for n sites in a distributed database. ODAM (Optimized Distributed Association Rule Mining) algorithm for distributed data association rule mining proposed by [83]. After discovering the global frequent-1 itemsets, it removes the infrequent ones and inserts the transactions and their count in a temporary file which is then used to find the frequent itemsets of larger lengths.

The focus of the algorithm is on the communication and synchronization. [19] proposed a parallel algorithm PFIN for mining frequent itemsets using data structure nodeset. It decomposes a large problem into small tasks executed in parallel. It is using a hash-based load balancing strategy for optimize resources.

DDM plays important role [11] in bioinformatics where database is heterogeneous in the distributed environment. A grid can provide accurate representation of available resources[14] and effective computational support [80] [43] for DDM where users have options to choose the resources and it is future of DDM [12].

As the XML system is more complex [69], DDM is more challenging. In data mining, rough set theory is a powerful approach [21] [22] which can be used for decision making. Frequent closed enumeration table (FCET) structure can save storage space and speed up producing the generalized association rules. The information is collected through e-CRM system which is mined by DDM and used for business decisions. ODAM algorithm proposed by [76], which is improved version of Apriori and based on CD and FDM algorithm. In which candidate support counts are not generated from the raw data set after the first pass. It removes infrequent itemsets, and places new transactions into memory. It reduces the transaction length and also data set size significantly.

Performance of any algorithm [76] depends on the number of nodes in the distributed setup. Execution time improves with increase of number of nodes or transactions. When number of nodes increases but number of transactions are less, algorithms take longer execution time [31]. FDM optimised with FP-Growth and DiffSet-data structure [31], a vertical data representation, improves the performance of FDM algorithm by reducing the memory requirements. [36] proposed algorithm based on “optimized matrix computation for multiparty data computation” which has some

challenges. Distributed data mining also helps [70] in maintaining privacy, reducing transmission cost, and sharing resources like memory.

Approaches in Large-scale data analysis [8] are (i) MapReduce and (ii) Parallel DBMS. MapReduce is easy, more fault tolerant technique, but large performance penalty as compared to parallel DBMS. Using the map-reduce approach, many algorithms can be processed in distributed environment, like the MRPrePost [40] algorithm gives the processing of prepost algorithm on the Hadoop platform. Many parallel algorithms based on the map-reduced technique in distributed environment are proposed in literature,. MRPrePost [40] and Prepost [66] algorithms are based on map-reduce where first one gives the processing of prepost algorithms whereas second one is cloud implementation and apriori map-reduce. The distributed simulations are available and their acceptance is very much dependent on their methods and tools [6].

A framework DT-DPM (Decomposition Transaction for Distributed Pattern Mining) [9] proposed in literature, which integrates Density-Based Spatial Clustering of Applications and distributed computing represented, CPU multi-cores and Single CPU for solving pattern mining problems. Bin Liu [14] proposed a DDM architecture based on the grouping similar data source to reduce communication. Its focus is to “improve the openness, cross-platform ability, and intelligence of the DDM system”. [75] proposed hybrid architecture by combining parallel and distributed mining which reduces redundant database and improves the performance. A framework is developed by [49] to analyse and manage the supply chain risk and gives better insight for better decisions.

Based on the literature survey, some of the popular distributed algorithms, issues and challenges, some basic terminologies, available tools and techniques, data sources in the field of distributed data mining are explained below.

2.1.1 Basic Terms of Frequent Pattern Mining Algorithm

- *Transactional Dataset* - Dataset having two columns, named, unique transaction ID and set of k-items.
- *Itemset* - The sets of items that appear frequently in the transactions of the dataset. If $I = \{I_1, I_2 \dots I_n\}$ be a set of distinct items in the dataset DS. Itemset is a set of items, A subset of I. An itemset A with k distinct items is referred as k-itemset [37].
- *Support* - Frequency of occurrence of an itemset in the transaction. It is defined as the percentage of transactions in the dataset DS that includes both itemsets A and B. The support count A / B is defined as [35][62]
$$\text{Support (A} \rightarrow \text{B)} = \text{Support (A} \cup \text{B)}$$
- *Minimum Support Threshold* – Association rules are considered interesting if they satisfy minimum support threshold which is set by users or experts. If an itemset satisfies the minimum support threshold, then it is frequent.
- *Local Frequent Itemset* - The frequent itemset generated from the local partition of database available on a site, having support count more than the local support threshold value is local frequent itemset.
- *Global Frequent Itemset* - The frequent itemset aggregated from local itemsets representing the entire database and having support count greater than the global support threshold is global frequent itemset.
- *Confidence* - The confidence is relationship between two items. It is the percentage of transactions in the dataset DS with itemset A that also contains the itemset B. It

is calculated using the conditional probability which is expressed in terms of itemset support. [35][62]

$$\text{Confidence}(A \rightarrow B) = \text{Support}(A \cup B) / \text{Support}(A)$$

Here, Support (AUB) is the number of transactions that contain both itemsets A and B, and Support (A) is the number of transactions that contain itemset A.

- *Association rule* - Finding interesting relationship or correlation amongst the items in the dataset with n number of transactions containing a set of items. An association rule is represented by A/ B, where A and B are distinct itemsets [62].

2.2 DATA SOURCES

There are many real and synthetic datasets available for the researchers in the area of data mining. These datasets are transformed and stored at different education or company sites working in these areas. Some of the popular data sources are listed below-

- <http://fimi.ua.ac.be/data/> :Actual Data sets [30] for data mining are available for research donated/ submitted by different organisation.
- <http://lib.stat.cmu.edu/datasets/> :Provides datasets collected on actual bases by different experiments by researchers, industry in the vast deversified areas.
- <http://rdatamining.com/> : There are many datasets available online for free for research use for different analysis.
- <https://pslcdatashop.web.cmu.edu/> : Pittsburgh Science of Learning Center Data shop. A central repository for research data.
- <http://almaden.ibm.com> : Produce synthetic datasets. Used by researches [16]

- <https://www.ics.uci.edu/ml/> : This is an online repository of large data sets which encompasses a wide variety of data types, analysis tasks, and application areas. Currently maintain more than 280 data sets for research. Researcher Ailing Wand used dataset form this repository. It maintains datasets for variety of applications including data mining.

2.3 DATA MINING TOOLS AND SOFTWARE

There are many software tools are developed for the different techniques of data mining with different feature and facilities. These tools are to be used based on the distributed environment suitable for the problem.

Table 2.1 Data Mining Tools & Software

Mining tool	Features	Techniques/Tools
Weka [25]	<ul style="list-style-type: none"> • Based on Parallel & Map Reduce • Memnory limitation with large dataset • Visualization • Open source • Good for developing new machine learning schemes 	<ul style="list-style-type: none"> • Association, Clustering, Classification, Regression, Pre-processing
Apache Hadoop [81] [78]	<ul style="list-style-type: none"> • Single mining task is split into many small tasks. • Each task runs on on one or many computing nodes • Batch-oriented parallel computing model • require Sun JDK 	<ul style="list-style-type: none"> • Based on MapReduce
RapidMiner	<ul style="list-style-type: none"> • Open source data mining tool • Transparent data handling • Scripting language based on XML 	<ul style="list-style-type: none"> • classification, regression, deviation detection, clustering, association, sequential Pattern analysis

	<ul style="list-style-type: none"> • Run on Every major platform 	
Oracle Mining	<ul style="list-style-type: none"> • operate on relational tables • multidimensional data analysis 	<ul style="list-style-type: none"> • Association, Clustering, Classification, Prediction, Regression, deviation detection
IBM SPSS Modeler / Clementine	<ul style="list-style-type: none"> • visual programming or <i>data flow</i> interface • Commercial software • Graphical interface • can access both structured and unstructured data 	<ul style="list-style-type: none"> • Association, Clustering Classification, Prediction
IBM Intelligent Miner	<ul style="list-style-type: none"> • Tightly integrated with IBM DB2 DBMS • Scalable • Visualization 	<ul style="list-style-type: none"> • Association, Clustering, Classification, Prediction, Regression, sequential Pattern analysis
MOA	<ul style="list-style-type: none"> • large data mining • open source software • use simple XML 	<ul style="list-style-type: none"> • classification, regression, clustering and frequent item set, frequent graph mining
Apache Spark[65]	<ul style="list-style-type: none"> • Distributed computing framework • Open source cluster computing • In memory parallel execution 	<ul style="list-style-type: none"> • Large scale data processing • Association, Clustering, Classification, Prediction, Regression, sequential Pattern analysis

2.4 ASSOCIATION RULE MINING ALGORITHMS

Association Rule Mining (ARM) algorithms can be classified as sequential, parallel, distributed, grid, and cloud based ARM. Initially many ARM algorithms have been proposed in the literature. These algorithms run on a single machine with a centralized database in sequence. Sequential ARM (SARM) algorithms are based on three different techniques namely Apriori [60], FP-Growth [34] and Eclat [45] and Clique. Many variant and updates of these algorithms are also proposed in literature like partitioning, sampling, H-mine, Eclat [45], FIN [85] etc. The size of the data generated is

increasing with time and single CPU and memory cannot handle the huge data and intensive computation load. Parallel ARM (PARM) algorithms are proposed to overcome these limitations, with ability to scaleup and to speed up the mining tasks. PARM [51] environment is tightly coupled system based on type of parallelism – task or data, architecture – distributed or shared memory, type of load balancing approaches, data layout – horizontal or vertical partitioning etc. PARM is ideal for centralized large database systems. Some of the PARM algorithms proposed in literature [68] are CD, DD, Can.D, DP3, sampling and other algorithms based on these algorithms.

For the last more than two decades distributed ARM (DARM) is gaining ground with the increase of multiple distributed datasets in place of centralized dataset and it is essential to use DARM for distributed datasets [51] where it deals with the loosely-coupled system where nodes situated in various sites are connected via LAN or internet. In this setup message passing mechanism is used for communication between different nodes. DARM provides the scalability, efficiency without the limitations of the centralized system. DARM are based on some traditional sequential algorithm and parallelism is achieved. The DARM algorithms are proposed in literature which use two approaches based on Apriori or Map-Reduce. Some of the DARM algorithms are FDM [23], PDM, CD [61], DD[60], ODAM [83], AprTidRec [4], LMatrix [32], PDF, PFIN [19], PPDM [46] and many more. In the recent years most of the DARM algorithms proposed are parallel in nature on MapReduce technology or mining on big data.

Some of the DARM Algorithm proposed in literature for mining association rules in distributed data environment are illustrated below: -

- *Fast Distributed Mining of Association Rules (FDM)* [23]: This algorithm is based on the Apriori algorithm. Two categories of DDM algorithms are Data Distribution

(DD) algorithm and Count Distribution (CD) algorithm. Apriori algorithm used at each site produces each length frequent itemsets and candidate set. FDM generates less number of candidate sets, and messages and it performs better than the standard sequential algorithms like CD and Apriori. There are some overheads of Parallel Virtual Machine and it needs more synchronization. FDM doesn't perform well with skewed organization of data.

Variants of FDM are also proposed in literature. FDM-LP[23] uses local pruning, FDM-LUP explores local as well as upper bound pruning whereas FDM-LLP explores local and polling site pruning.

- *Distributed Mine closed Itemsets (DMCI)* [17]: Distributed Frequent Closed Mining Algorithm finds the local frequent itemset, communicate with other site to collect their support, perform local pruning to generate global Frequent Closed Itemset (FCI) where FCI is a small subset of frequent itemset where redundancies are discarded and it is more concise and meaningful. The communication load between different sites is small, but there is frequent communication between sites. FCI mining provides help to take correct decision at each site, but time consuming, several data scans.
- *Distributed Mining of Association Rules (DMA)* [24]: The mining steps in this algorithm are candidate sets generation, local pruning and message optimization. The algorithm generates less candidate sets at local sites as compared to Apriori-gen. Optimization technique eliminates the duplication from the candidate sets. It determines the polling site or host site for broadcasting, collecting support counts and determine whether X is large for each candidate set X using hash function. It does not require to scan the partition again to calculate the support counts. It shows

that DMA performed better when the number of nodes is higher. It requires more Storage for message and support counts. It uses the nodes having identical schemas only.

- *Distributed Mining Association Rules with Item Constraints (DMCA)* [18]: The algorithm integrates the item constraints into the mining process which can acquire more efficient algorithms. The candidate sets are reduced, and execution time improves as compared to FDM. The item constraints are formalized with boolean expression which is used for generating candidate sets satisfying the constraints. The algorithm uses the relationships between global and local frequent itemsets and saves the local counts in hash tables. It reduces dataset scans and communication cost.
- *Distributed Decision Miner* [10]: It is based on the Apriori and CD algorithms. It is good for un-skewed data. It verifies that an itemset is large before collecting its support counts from all nodes. The algorithm is scalable and data skew robustness with low communication overhead, but it requires more storage.
- *Multi-Relation Data Mining (MRDM)* [79]: Algorithm is based on the Genetic (GA), Apriori and extended Algorithms. It combines multi-relations using genetic algorithm. The steps are (i) GA to mine antecedent rules, (ii) mine the consequent rule from the rest relational attributes of other tables (iii) produce extended association rule using foreign key. It produces finer patterns and contains more information. This is suitable for small databases.
- *CREDLM & CREMLM Algorithms* [16]: Chain Retail Enterprise based on Massed Learning (CREMLM) algorithm refers to the large-scale chain retail enterprises

with large bandwidth and small transaction on each branch store with many branch stores. It is performing the globally frequent prune at one store so called massed learning. The communication frequency is low with complexity $O(n)$ only. It requires high data scans, low local efficiency. Performs better with minimum support is high.

Chain Retail Enterprise based on Dispersed Learning (CREDLM) algorithm [16] refers to the small-scale chain retail enterprise with small bandwidth and large number of transactions on each branch store with few branch stores. It is performing the globally frequent prune on many stores so called dispersed learning. Communication load impact the performance of algorithm. CREMLM has performed better than CREDLM in a larger bandwidth whereas it is reversed in smaller bandwidth. Reducing the Data scanning may improve the performance of these algorithms.

- *AprTidRec Algorithm* [4][5]: This is an extension of the Apriori algorithm. A record structure “tidrec” is defined to store each candidate frequent itemset from the ordinal distributed data and join is used in place of pruning. It concluded with short time complexity and reduced communication cost. The trade-off between time and space complexity is needed to get the best results. The same can also be applied to non-isomorphic data sources.
- *Count Distribution (CD)* [61]: CD algorithm[41] is parallelization the Apriori algorithm with horizontally fragmented dataset. The candidate sets generated at each site is broadcasted to every other site. It exchanges only counts between sites and message exchange is $O(n^2)$. Memory not properly utilized by the algorithm and it is not much scalable.

- *LMatrix* [32]: Algorithm creates the local LMatrix, an object-by variable compressed structure, to calculate local support which saves time to scan the database partitions but consumes more memory. A pre-fix tree structure FP-tree is used to store compressed information about frequent itemsets. FP-array technique is used to improve the performance for FP-tree based for sparse datasets. It is useful when the dataset is too large for sequential mining. The algorithm reduces the communication cost, size of messages, time to scan the partition. It consumes more memory and execution time increases when data size increases and minimum support threshold decreases.
- *Frequent Pattern Growth (FP-Growth) algorithm* [34]: FP-Growth is an improvement of apriori algorithm. This algorithm finds the frequent itemsets in the database without generating the candidate sets. FP-growth constructs a compressed tree like structure called FP-tree to represents frequent itemsets. It is extension of prefix-tree structure which provides a technique to use compact tree data structure for data mining to reduce data scans. FP-tree is expensive and may not fit into the memory.
- *R-Apriori (Reduced apriori)* : R-Apriori is a parallel ARM algorithm proposed by Rathee [65] based on Apriori algorithm on horizontal partitioning of dataset. It is a parallel algorithm on spark platform for the MapReduce framework. It is not mining exact association rule but it mines approximate association rules. It reduces the computations in second iteration and faster than Apriori. Performance improves with the increase of data size.
- *Optimized Distributed Association Rule Mining (ODAM)* [83]: It is an improved version of Apriori algorithm. It considers the horizontal partitioning of dataset. It

minimises the candidate sets generation so it offers better performance. It sends support counts to one site reducing communication. Rules are generated by different participating sites are identical ensures synchronization.

- *Parallel Frequent Itemset Mining Algorithm Using Nodesets (PFIN)*: PFIN is a parallel FIM algorithm [19] based on the efficient data structure Nodeset, implemented on spark framework. It breaks a large scale problem into small tasks and run them parallel. It groups the frequent itemsets and uses hash based load balancing technique. It has low communication overheads.

2.4.1 Data Structures Used in Data Mining

Data Structures act an important role in DARM. Efficient data structure reduces the computational complexity of an algorithm which makes it better. Different data structures are proposed in the literature and based on these data structures different DARM algorithms are proposed. Using tree based data structures reduces the data scans and stores information about the frequent itemsets generated in the frequent patterns mining. Some of the popular data structures are given below-

- *Frequent-pattern tree (FP-Tree)* is a popular data structure used by the FP-Growth[34] algorithm. It is a compact extended prefix-tree structure for storing frequent patterns. Database is scanned twice for the construction of the FP-tree. It is an improved version of bidirectional prefix tree structure that allows bottom up scanning. For large pattern, FP-tree construction complexity is very high and expensive.
- *Nodeset data structure* used for mining process of frequent itemsets is based on POC tree proposed by [85] and used in FIN algorithm for sequential association rule mining in 2014. Nodeset is an efficient data structure[85] which require less memory. It stores

only postorder or preorder of nodes in the form of N-info. The database is scanned only once and tree is constructed with one root and subtree as children.

- N-list is a vertical data representation compact data structure proposed in recent years. N-List is a structure like FP-tree [87] and stores information obtained from PPC-tree about the itemsets using preorder, postorder. It has been proven to be very efficient for mining frequent itemsets.
- Trie data structure is a prearranged data structure proposed by [29]. It is also known as radix tree, digital tree or prefix tree. In Trie structure strings are used to store the keys. The trie structure is traversed depth-first, following the links between nodes, which represent each character in the key.

2.4.2 Issues And Challenges

Issues in designing a DDM algorithms for mining association rules in the distributed data environment are to be addressed which are evaluated due to the distributed nature of the database. On the one side we are getting the advantage of parallel processing, scattered storage requirement at different sites by reducing the load on the centralized resources but on the other side there is a need to evolve techniques to find local frequent itemsets that can be used to find the global frequent itemsets. It requires (i) local processing capabilities and (ii) storage at all the sites, (iii) more data scans (iv) a lot of communication between these sites (v) communication with centralized or main site, (vi) synchronization, (vii) merging the information etc. Algorithms for DDM for association rules mining proposed in literature are addressing these issues. It is to be considered that reducing one cost may increase the others. So, algorithm addressing these issues and improving costs and performance etc., keeping trade-off between them, is required to suit a particular

distributed scenario. Some of important distributed data association rule mining issues are:-

- *Number of data scans*: Number of data scans [67] require to find frequent itemsets and support count and it effects the performance.
- *Number of candidate sets*: Candidate sets are generated for finding frequent itemsets. Reduce number of candidate sets will reduce the communication overhead.
- *Communication cost*: Sites communicate FI and support counts. Reducing the number of messages between nodes will reduce the communication cost.
- *Memory requirement*: Optimum use of the memory is required to store the local and global FI, transactions, support counts etc.
- *Pruning* : Prune out not frequent candidate sets to reduce the number of frequent itemsets which may not be frequent hence reduce the communication load.
- *Synronisation*: Needs between the nodes and centralised site.
- *Work load balancing at various nodes*: One or all nodes participate in computing frequent itemsets.
- *Rule redundancy* : Some of the itemsets are frequent at more than one node.
- *Scalability* : Some of the algorithms are good for small database only and do not scale-up well.
- *Skewness*: Data skewness[44] may improve the performance by reducing communication. It can also have adverse effect of load imbalance.
- *Data decomposition among nodes*: May reduce the candidate sets and redundancy.
- *High Dimensionality*: Handling the increase of number of dimensions [50] or items

These issues and challenges are not independent of each other. There are

relationships between these issues in the sense that some are directly proportional and some are inversely proportional to others given in Table 2.2 and 2.3. Based on the size of the database, number of partitions, communication bandwidth availability, processing power and available recourses, trade-off between them are required.

Table 2.2 Inversely proportional issues

Number of data scans	Memory /storage requirement
Skewness	Pruning
Pruning	candidate sets
Work load balancing	Communication, skewness

Table 2.3 Directly proportional issues

Number of candidate sets	Communication
Communication	Synronisation
Rule redundancy	Communication
Good Data decomposition among nodes	Candidate sets, redundancy,

2.5 APPLICATION MODELS OF DARM

Applications of association rule mining [3] in large and dispersed database are businesses, defence, public safety, GIS, medical diagnosis, Hospital etc. As the database is updated on regular basis, so transferring and storing data at one centralised place is not feasible and time consuming, and mining data must be up to date [56] otherwise it affects the decisions. Distributed computing is important in data mining as mining requires huge amounts of resources and data distribution is required for safety, scalability and resource sharing [7]. Lot of mining application models and algorithms are developed and used effectively for the decision making, expansion and making good strategies by different

business houses. The most popular algorithm for association rule discovery from market, sales databases [7] is Apriori algorithm. This algorithm performance is good with sparse datasets. Association rules are used to examine the customer buying behaviours for different item purchased from the supermarket transaction data. It describes how often the items are purchased together[34]. Researcher used uninorms[63] to aggregate support and confidence in market basket analysis for UK grocery. Knowing the sales pattern by the customer can help to make potential strategies to increase the sales and recommendations[82].

The main purpose of the research conducted by [71] on Market Basket Analysis of beauty products was to find the interrelation of the products in a beauty shop and use the outcome for marketing and sales promotion of the products. The researcher used the Apriori algorithm for this research. It is observed that there are two type of customers, casual and regular, where first who buy a few random items and second who planned and buy more. The support and confidence value used in this research were 0.1 and 0.4 respectively [52]. A case study on analysing market basket was conducted by [2] on an electronic store data using FP-Growth[34] and Apriori algorithm and generated useful rules. [38] also performed research for customer basket and found items with highest correlations. Sales Analysis is performed by [64], same as market basket analysis to target prospective customers for the optimizing the profit. The work used product rating in data mining Techniques. [1] applied the association mining in finding association between different parameters causing the road accident. This study is useful for finding hidden association and improving the road safety.

Liu [15] proposed DDM model “DRHPDM - Data source Relevance-based Hierarchical Parallel Distributed data mining Model” for e-business. Its focus was to

“improve the openness, cross-platform ability, and intelligence of the DDM system”. Norulhidayah [54] proposed applied DDM on daily sales analysis on the data of Chan furniture and proposed a strategy for business promotion and marketing. A periodic mining, due to the dynamic nature of data in the market basket data, is proposed [48] by automate the threshold values by working on the change modelling concept. Study is conducted by [39] and a technique is proposed to find associations among items and determine better showcase to appeal to customers and increase sales. Map-Reduce based Consistent and Inconsistent Rule Detection (MR-CIRD) algorithm is proposed [26] to detect the consistent and inconsistent rules from big data and provide useful and actionable knowledge to the domain experts. These pruned interesting rules also give useful knowledge for better marketing strategy as well. Merouane [47] presented a model for distributed association rules mining by combining it with the multi-agent system to run it in parallel and distributed manner from the centralized ERP database to provide perfect response time. There are many risks also in the supply chain mining due to the different technologies used. There are some negative associations[55] along with the positive associations that exist in the data which are very significant for the business. Negative association is expressed as if there are no common transactions between any two positive regular itemsets, then they could be declared as negative regular item set or contradicting patterns. A framework is developed by [49] to analyse and manage the supply chain risk which gives better insight for better decisions. There are a lot many untouched areas where application of data mining can be explored in future.

Types of Frequent Patterns

- Itemset[35]

Example: Bread, Egg and Bread, that appears frequently together in a transactional

dataset of Market Basket Analysis.

- Subsequence[35]

Example: Buy products in specific order like, buy computer then buy speaker in a shopping history.

- Substructure[35]

Example: Subgraph and Subtree

Major Mining Applications

- Market basket analysis [35]
- Estimation of financial records
- Retail business
- Telecommunication business
- Biological data scrutiny
- Data mining for invasion innovation
- Data mining in other controlled applications
- Tourism Industry

2.6 RESEARCH GAP

Based on the extensive literature review in the area of association rule mining, algorithms and techniques, it is observed that the distributed data association rule mining is less explored. Most of the research in the area of association rule mining is done on the centralized database. Some of the research uses the parallel approach by distributing data amongst the sites and perform parallel processing in order to reduce the load on a centralized processing capabilities. There is very little research which considers the data

already distributed to various sites and the data partition available at sites vary in size. Recently some of the new data structures like N-list, Nodeset are proposed for the association rule mining but not implemented for the distributed data as such. Due to the increase of globalisation and generation of huge data at different locations there is a need to explore the applicability of the mining algorithms of the purely distributed data.

3 METHODOLOGY

The problem of mining association rules is divided into two subproblems[62]: (i) to find all frequent itemsets in the database for the given minimum support threshold value, and (ii) to generate the association rules using the frequent itemsets found in (i). As the mining association rules cost is mainly involved in (i), the focus is on the evolution of some efficient technique for the first subproblem [62]. The proposed algorithm are also focused on finding the global frequent itemsets and then an application model is proposed for finding the association rules. This chapter explains the DARM process and methodologies used in development of the proposed algorithms.

3.1 DISTRIBUTED DATA MINING PROCESS

The DARM process involves a lot of activities to be performed by the algorithm at different sites to mine the association rules from the distributed data. These activities are not disjoint. In order to perform the mining on a distributed data, some assumptions are taken into consideration.

3.1.1 Assumptions

- Database is horizontally partitioned and distributed at various sites around the globe.
- Data is generated or captured at different sites.
- Data is not transferred between nodes due to the resources constraints or policies.
- Size of each partition i.e., number of transactions stored at each site may differ.
- Number of candidate sets generated at each site may differ.

- All sites may not be equally loaded.
- The resources and capabilities are similar at various sites.

3.1.2 Process Activities and Definition

- *Data scanning* - scanning transactions in the database to compute frequent itemset
- *Local frequent itemset (FI)* - The itemsets that meet a user-specified minimum support at each node. A frequent itemset is maximal if it is not subset of any other frequent itemset.
- *Data storage* - Storage is required on each site for storing local support counts, transactions, frequent itemsets etc.
- *Skewness* - Distribution of itemsets among the various partitions. A database's total data skewness is the sum of the skew of all itemsets weighted by their supports. Most algorithms require low data skewness for good load balancing.
- *Local pruning* - Removing the infrequent itemsets from candidate sets at each node. It reduces number of candidate sets, cut communication cost.
- *Candidate set generation* - Candidates are generated in parallel. Each processor generates its own local set.
- *Count polling* - A polling site is assigned to itemsets X independent of the site where X is locally large.
- *Polling site* - responsibility to find whether X is globally large. It broadcasts requests to collect local support counts for X and compute global support count.
- *Communicating Local FI* -Communication between sites or communicating site with other sites sending FI by message passing.

- *Global pruning* -Additional pruning is being done to global frequent itemsets. The efficiency of local pruning can be enhanced by global pruning.
- *Finding Association rule* - According to the global frequent itemsets and minimum support, association rules based on the global database can be acquired.

The DARM definition is given below as taken from [23] is defined below:

Let DB is a transaction database with $I = \{i_1, i_2, \dots, i_m\}$ set of items. Transaction T of DB is a set of items where $T \subseteq I$. An itemset $Z \subseteq I$, belongs to T if and only if $Z \subseteq T$. An association rule (AR) is represented[23] as $Z \Rightarrow Y$, where $Z \subseteq I$ and $Y \subseteq I$ and $Z \cap Y = \phi$. The AR $Z \Rightarrow Y$ holds in the database with a confidence 'c' implies that the probability of a transaction in database containing Z also contains Y is 'c'. The association rule $Z \Rightarrow Y$ has support 's' in database implies that the probability of a transaction in database contains both Z and Y is 's'. The association rules mining is a task to search all the association rules in the database where support is greater than the minimum support threshold value and confidence is greater than the minimum confidence threshold value.

For an itemset Z, support is defined as the percentage of transactions in database containing Z, and its support count, Z.sup, is total number of transactions in database containing Z. An itemset Z is large or frequent occurring if its support is equal or greater than the minimum support threshold. An itemset of size k is called a k-itemset. Distributed algorithm[23] for mining association rules statement.

To examine the association rules mining in a distributed database DB with D transactions and n-sites S_1, S_2, \dots, S_n having n-partitioned $\{DB_1, DB_2, \dots, DB_n\}$ respectively. Let D_i be the size of the partitions DB_i where $i = 1, 2, \dots, n$. Z.sup, the support counts of an itemset Z in database and $Z.Supp_i$ in DB_i . For each site S_i , $Z.Supp_i$ is

the local support count of Z and $Z.\text{sup}$ is the global support count. For a specific minimum support threshold value 's', Z is also globally large itemset if $Z.\text{sup} \geq s \times D$; correspondingly, Z is locally large itemset at site S_i , if $Z.\text{sup}_i \geq s \times D_i$. Let L be the globally large itemsets[23] in database, and $L_{(k)}$ the globally large k-itemsets in L . A distributed association rule mining algorithm finds the globally large itemsets L .

3.2 DISTRIBUTED DATA MINING MODEL

Many DARM architectures are proposed in the literature. The most suitable architecture for the distributed data mining is given in Figure 3.1, which was proposed by [27] as a general purpose DARM architecture. It consists of distributed data at various sites where local mining is performed and then these are aggregated to the global model. It is suitable from small as well as large scale distributed system.

Proposed model with distributed data collection and maintenance at various sites without any centralized data repository and without any centralized data mining unit is given in the Figure 3.2. There is no centralized database and no centralized data mining unit. Data is partitioned and distributed amongst various sites, where data is collected and stored. Local mining is done at various sites and frequent itemsets are found. Coordinating unit assigns the polling site for the global processing. Number of processing units in coordinating unit may vary depending on the load based on the size of the overall data and number of sites. There is no centralized processing unit to process the global mining rather all sites participate in global mining. This is best suitable model for a large setup and large data size with a good amount of load distribution amongst the available resources.

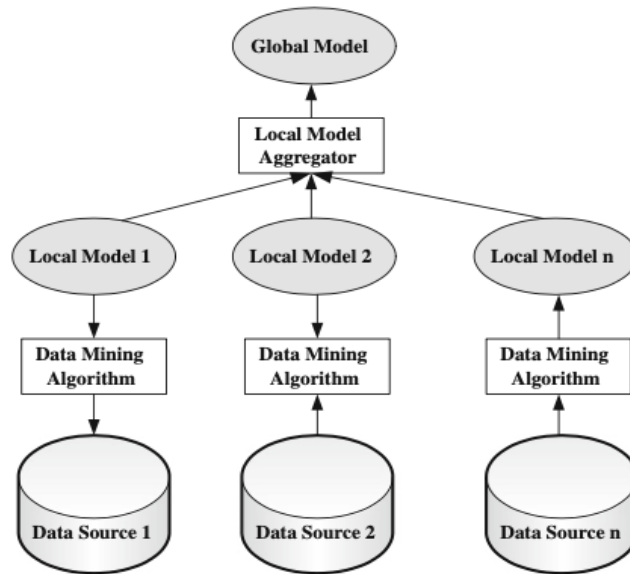


Figure 3.1 A general purpose DARM architecture [27]

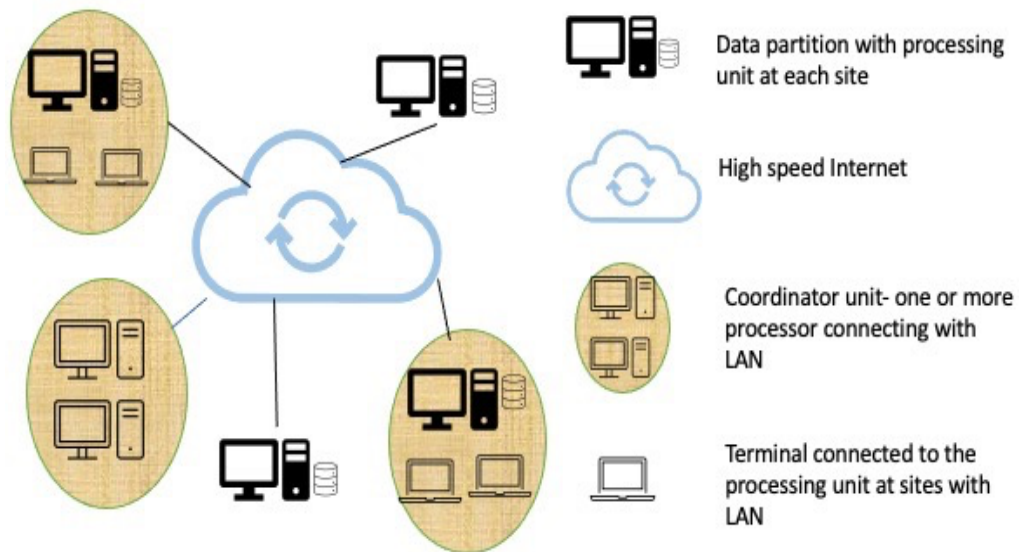


Figure 3.2 Proposed model for DARM

3.3 CANDIDATE SET REDUCTION BY PRUNING

Pruning is a process of reducing candidate sets[23] generated by data scan for itemsets size $k=1,2,..n$. It eliminates the frequent itemset which are not locally large

itemset i.e., having support count less than the minimum support threshold as those may not be the global frequent itemsets. This reduces the number of candidate sets for communication to other nodes so reduces the communication load over the network and enhance the performance.

Example:

If frequent size-2 itemset at site -2 {ab, ac, bf, cf}.

After pruning itemsets having support count less than the minimum support threshold say {cf, bf} are removed from the candidate sets.

The remaining candidate sets {ab, bf} communicated to all other sites.

3.4 NO BROADCASTING OF FREQUENT ITEMSETS

Broadcasting of the local frequent itemsets to all sites is a heavy load on the communication network. No-broadcasting reduces the load on the network. All local frequent itemsets are sent to a dedicated or coordinating site for assignment of polling site for finding the global frequent itemsets. There is no-broadcasting of the local frequent itemsets by different sites.

Example:

Suppose there are 5 sites then all sites send frequent itemsets to other four sites means $5 \times 4 = 20$ packets but in no-broadcasting all 5 sites send local frequent itemsets to one site only so $5 \times 1 = 5$ packets are being sent on the communication channel. With the increase of the number of sites there is a big reduction in the packets communicated over network in the no-broadcasting technique and the performance of the algorithm improves. Total number of FI messages send are of $O(n)$ only

3.5 NODESET DATA STRUCTURE

Nodeset data structure[85] is a latest, novel and efficient data structure proposed for data mining and it is based on the pre-order coding tree called POC-tree.

3.5.1 POC Tree

POC tree is constructed with one root and prefix subtrees. Database is scanned for frequent size -1 itemsets and their support count. 1-itemset are arranged as per their support count in descending order. Create the root of a POC-tree labelled as “null” For each transaction in data partition. If Tree has a child M such that M.item-name = item-name, then increase M’s count by 1; Otherwise create a new node M, with its count set to 1, and add it to Tree children-list. If list is nonempty, insert tree recursively. Scan the POC-tree to generate the pre-order of each node by the pre-order traversal. This is an efficient data structure, which use POC tree and reduces data scans and increase efficiency.

Table 3.1 Sample Database Transactions

TID	Items	Ordered frequent items
101	b, e, j, i, p	b, i
102	f, c, b, i	b, c, f, i
103	b, c, h,	b, c
104	f, a, b, c	a, b, c, f
105	a, f, c, g, b	a, b, c, f

Ordered frequent items are items having support count \geq threshold

Here support count threshold is considered as 25% and frequent itemsets available in each transaction are identified and ordered, shown in Figure 3.3.

N-info – pair of preorder and count

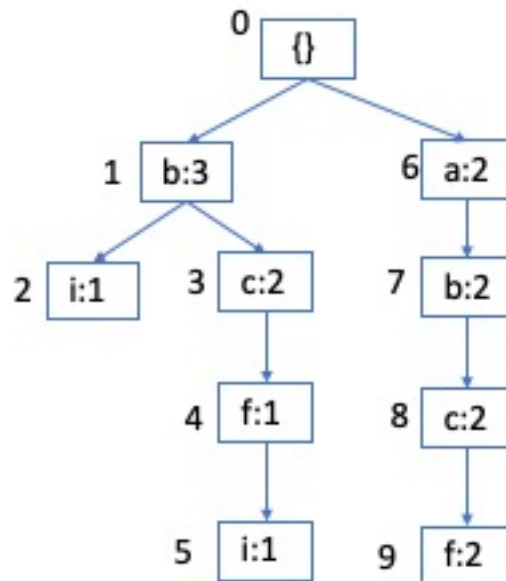


Figure 3.3 POC Tree construction

3.5.2 Nodesets

Nodeset of frequent item-*i* is sequence of all n-infos of node *i* in the POC-tree. Each N-info of a node in the POC-tree is count of the number of transactions with item *i*. Therefore, *i*'s support is the sum of counts of nodes with item *i*.

Nodeset of frequent itemset & support count

$b \rightarrow \{(1,3), (7,2)\}$	5
$c \rightarrow \{(3,2), (8,2)\}$	4
$f \rightarrow \{(4,1), (9, 2)\}$	3
$i \rightarrow \{(2,1), (5,1)\}$	2
$a \rightarrow \{(6,2)\}$	2

Nodeset of frequent 2-itemset & support count

The Nodeset of 2-itemset is a subset of i_2 's Nodeset

bc	$\rightarrow \{(3,2),(8,2)\}$	4
bf	$\rightarrow \{(4,1),(9,2)\}$	3
bi	$\rightarrow \{(2,1),(5,1)\}$	2
ac	$\rightarrow \{(8,2)\}$	2
.....		

Nodeset of frequent 3-itemset & support count

abc	\rightarrow nodeset $ac \cap$ nodeset of bc	
	$\{(8,2)\}$	2

3.6 ZERO-FIRST TECHNIQUE

In the real life scenario database is distributed where data is captured or gathered at different locations. The size of the data partitions varies[9] in size from a few hundred of transactions on one site to a million of transactions at other site. In the distributed data mining, resources are also distributed and there should be a mechanism to utilize all the nodes by allocating processing to less occupied nodes. In this work a new technique for load balancing Zero-first for distributed data mining is presented.

Based on the assumptions, Zero-first technique is developed for the assigning polling sites to each local FI received by the coordinating site from all other sites. The polling site is responsible for finding the globally large itemset from the list of locally large itemset. The new technique ensures that the load is assigned to less occupied sites for a distributed data association rule mining.

Definition: Zero-first technique:

For sites $S = \{S_1, S_2, \dots, S_n\}$ and locally large candidate sets $\{CG_1, CG_2, \dots, CG_n\}$ received from n sites.

$$\text{Candidate set } CG = \{CG_1, CG_2, \dots, CG_n\}$$

$$\max_CG = \max\{CG_1, CG_2, \dots, CG_n\}$$

Where \max_CG is the max number of candidate sets broadcasted by any site.

if $\max_CG > \text{anyone}\{CG_1, CG_2, \dots, CG_n\}$

for $CG_i < \max_CG$ Arrange

$$S' = \text{Order}\{S_1, S_2, \dots, S_{n-q}\}$$

otherwise $S' = \text{Order}\{S_1, S_2, \dots, S_n\}$ all sites

Sites are arranged in the order of size of the candidate sets generated, starting with zero, excluding the sites with maximum size of candidate set.

If the candidate sets sent by each site are not equal, then all the sites are to be arranged in the order of the number of candidate sets generated by each site.

CG' = complete combined list of all locally large itemsets received from all the sites removing duplicates

$$CG' = \{CG_1 \cup CG_2 \cup \dots \cup CG_n\}$$

Allocate CG' to sites S'

After arranging the sites in order of their loads, allocation of polling sites to the itemset are done in the order of the site occupancy (zero-first order) to check it for globally large itemset. [23] If all the frequent itemsets are not assigned then repeat the assignment in the same sequence of initial assignment.

Example:

Let there are four sites S_1, S_2, S_3, S_4 . The candidate sets broadcasted by site $S_1\{\}, S_2\{ab, bf\}, S_3\{ab\}, S_4\{bc\}$. Applying zero-first technique, the ordered candidate set is $\{ab, bc, bf\}$ and ordered site set is $\{S_1, S_3, S_4\}$ leaving most occupied site $\{S_2\}$.

Using the zero-first technique, the first polling site is $\{S_1\}$ and first locally large itemset in the ordered candidate set $\{ab\}$ is assigned to site $\{S_1\}$. Similarly $\{bc\}$ is assigned polling site is $\{S_3\}$ and $\{bf\}$ itemset is assigned to $\{S_4\}$. Leaving the most occupied sites who sent maximum number of locally large itemsets.

Sites with small data partitions have fewer number of transactions are under-utilized in terms of the processing, memory and scan time as compared to the sites with large data partitions. The Zero-first technique assigns more responsibility to less loaded sites and does not assigns any load to the most occupied sites. It ensures load balancing across the sites.

3.7 SIZE BASED ASSIGNMENT TECHNIQUE

In the distributed setup, data is generated or created at different locations. The number of transactions at various sites differ [9] from a few hundred to millions of transactions. In this setup, resources at each site is also limited and are distributed. Data mining requires great amounts of resources [7] so techniques for flexible distribution of work load amongst sites need to be developed. The sites with little number of transactions are less occupied as they require less memory, computational capabilities and time for maintaining data, scanning for frequent itemsets and maintaining candidate sets.

New technique SBA is proposed to assign the polling sites to the locally large itemsets received by a designated site. The polling site finds the globally large itemsets from the itemsets received. Novel technique takes care of the sites with the large data partitions and distribute load considering the number of transactions at each site and balance the load.

Definition: Size Based Assignment Technique:

For sites $S = \{S_1, S_2, \dots, S_n\}$ and candidate sets $\{CG_1, CG_2, \dots, CG_n\}$ sent by n sites.

Total Transactions $TT = \{T_1, T_2, \dots, T_n\}$

For size k-itemsets

Average Transactions percent per site, $AT = \frac{100}{n}$

Actual number of Transaction in percent at site

$$PS_i = \frac{T_i}{\sum T_i} \times 100$$

Load difference at site in percent S_i is $\Delta_i = AT - PS_i$

For all k-sized itemsets, complete candidate set

$$CG = \{CG_1, CG_2, \dots, CG_n\}$$

CG' = Complete Candidate sets received from all sites without any duplicates

$$CG' = \{CG_1 \cup CG_2 \cup \dots \cup CG_n\}$$

Average candidate sets at each site in percent

$$ACG = \frac{CG'}{n}$$

Arrange sites as per the partition size S'

Local frequent itemsets assigned to the polling site S_i is $= \lceil ACG + \Delta_i \times CG \rceil$

After finding the average frequent itemsets to be allocated to each site, from site list S' arranged in the order of their number of transactions, the assignment of polling sites to the frequent itemsets is done in this order by assigning upper integer value of average candidate set plus load difference. Assignment continues till the entire candidate set exhausted. This way the highly loaded sites are assigned nil or very little number of locally large itemsets for finding globally large itemsets. This technique assigns the load inversely proportional to the site load and hence balance the load of polling station.

Example :

Let there be five sites S_1, S_2, S_3, S_4, S_5 having transactions 10% , 15%, 20%, 25% and 30% of the total transactions respectively. The frequent itemsets sent by sites $S_1\{1, 4\}$, $S_2\{2, 4, 6\}$, $S_3\{3, 8, 17, 16\}$, $S_4\{12, 17, 1, 18\}$, $S_5\{2, 8, 19, 11\}$. Applying SBA technique, the ordered candidate set is $\{1, 2, 3, 4, 6, 8, 11, 12, 16, 17, 18, 19\}$ with 12 locally large itemsets and ordered site set is $\{S_1, S_2, S_3, S_4, S_5\}$ with average transactions 20% as there are 5 sites.

The load difference at S_1 is 10% so local frequent itemsets to be assigned to site S_1 is

ceiling integer $\left(\frac{12}{5} + \frac{10}{100} \times 12\right) = 4$ i.e. $\{1, 2, 3, 4\}$ items

Similarly assignment to S_2 is: ceiling integer $\left(\frac{12}{5} + \frac{5}{100} \times 12\right) = 3$ i.e. $\{6, 8, 11\}$ items

Assignment to S_3 is : ceiling integer $\left(\frac{12}{5} + \frac{0}{100} \times 12\right) = 3$ i.e. $\{12, 16, 17\}$ items

Assignment to S_4 is : ceiling integer $\left(\frac{12}{5} + \frac{-5}{100} \times 12\right) = 2$ i.e. $\{18, 19\}$ items

Assignment to S_5 is : Zero items as candidate list exhausts.

The sites with small data partition or with little number of transactions are not fully occupied. These sites have less processing scan and memory needs, as compared to the sites with more transactions. Proposed SBA technique assigns load inversely proportional to the site occupancy by considering the partition size and balances the load.

3.8 COMPARISON METRICS

The proposed algorithms are compared with the popular distributed data mining algorithms FDM [23] and PFIN [19]. FDM is one of the most popular distributed data mining algorithm and the PFIN is parallel algorithm using nodeset data structure and performed better as compared with FIN and PFP two latest algorithms where FIN is

sequential algorithm using nodeset data structure and PFP is parallel FP-Growth[34] algorithm.

Experiments are performed on a cluster of 4, 5 and 6 nodes connected with LAN, using Intel i5, 64 bit processor running @ 3.3 GHz with 6 GB RAM, 1 TB HDD, running on windows 10 OS, having Java JDK 1.7.

Functionality and performance of the algorithms are tested on four datasets including three real datasets mushroom, connect, chess and one synthetic dataset T10I4D100K. These datasets are available for research on data mining on the FIMI data repository (<http://fimi.ua.ac.be>) [30]. Mushroom dataset is created using attributes like shape, surface, colour, order, cap etc of 23 species of gilled mushrooms, connect and chess datasets are generated from different game steps, and T10I4D100K is a synthetic dataset generated using the IBM Quest generator. Datasets specifications are given in Table 3.2.

Table 3.2 Datasets and their specifications

Dataset	Total Trans.	Number of Items	Average Length	Type	File Size
Mushroom	8124	119	23	Dense	570 KB
Connect	67557	129	43	Dense	9.3 MB
Chess	3196	75	37	Dense	342 KB
T10I4D100K	1,00,000	1000	40	Sparse	4 MB

Support threshold values (in percentage) taken for study

Mushroom 10, 20, 30, 40, 50 and 60.

Connect 10, 20, 30, 40, 50 and 60.

Chess 10, 15, 20, 23, 30, and 35.

3.9 SUMMARY

This chapter explains the methodology used in the development of algorithms and application model in the area of distributed association rule mining. The chapter first gives the assumptions taken in development of the algorithms. Then it introduces the DARM process and explains the steps involved in the process. It then, describes the no-broadcasting technique for reducing the load on the communication network, and candidate set reduction by pruning techniques for finding the global frequent itemsets. The data structure used at each site is explained in detail. New techniques zero-first which allocates more polling sites to the sites with little number of candidates, is explained in detail. Another technique size-based assignment of the polling sites for assigning load to the less loaded sites is explained with an appropriate examples.

4 QUICK DISTRIBUTED FREQUENT ITEMSET MINING USING NODESET

In this chapter, new algorithm Quick Distributed Frequent Itemset Mining using Nodeset (QDFIN) is proposed which uses efficient data structure nodeset at each site and construct POC-tree [85]. It uses local and global pruning technique to reduce the candidate sets and zero-first technique for assigning polling site for load balancing. Assignment of globally large itemsets computation to the less loaded sites increase the computational capacity.

4.1 PROPOSED ALGORITHMS

Symbol Description[23]

s - Support threshold min-sup;

D - Number of transactions in database;

L_k – Globally large k-itemsets;

$Z.\text{sup}$ - Global support count of Z ;

CA_k – Candidate sets generated from L_k ;

D_i - Number of transactions in DB_i ;

$GL_{i(k)}$ – globally large k-itemsets at S_i ;

$CG_{i(k)}$ - Candidate sets produced by FIN algorithm;

$LL_{i(k)}$ - Locally large k-itemsets in $CG_{i(k)}$;

$Z.\text{sup}_i$ – Local support count of Z at S_i

$LP_{i(k)}$ – Local pruning k-itemset at site S_i

Based on the definition of POC tree given in section 3.2.1 of chapter 3, POC construction algorithm -1 is given below. The nodeset data structure using POC tree is used by each node for storing the frequent itemsets created for each data partition available at each node using the FIN algorithm proposed by Zhi-Hong et. al. [85]. The FIN algorithm is given below algorithm-2. Algorithm – 3 is the proposed algorithm of the zero-first technique for the assignment of frequent itemsets to the polling sites for generating the global frequent itemsets. Finally Algorithm - 4 is the proposed QDFIN algorithm for finding association rules in distributed data.

Algorithm - 4.1: POC-tree Construction

Input: A database partition and a minimum support s .

Output: A POC-tree and frequent 1-itemset $F1$ (the set of frequent 1-itemsets).

1. [Frequent 1-itemsets Generation]

Scan the data partition to find 1-termset and support count as per min-sup s

Arrange 1-itemset as per support count in descending order

2. [POC Tree]

Create the root of a POC-tree labelled as “null” For each transaction in data partition do Select the FI and arrange them according to $F1$.

Call insert tree. If Tree has a child M such that $M.item-name = item-name$, then increase M 's count by 1; Otherwise create a new node M , with its count set to 1, and add it to Tree children-list. If list is nonempty, call insert tree recursively.

3. [Pre-code Generation]

Scan the POC-tree to generate the pre-order of each node by the pre-order traversal.

Algorithm - 4.2 : FIN

Input: A transaction database DB and a minimum support ξ .
Output: F , the set of all frequent itemsets.

- (1) $F \leftarrow \emptyset$;
- (2) Call Algorithm 1 to construct the POC-tree and find F_1 , the set of all frequent 1-itemset;
- (3) $F_2 \leftarrow \emptyset$;
- (4) Scan the POC-tree by the pre-order traversal **do**
- (5) $N \leftarrow$ currently visiting Node;
- (6) $i_y \leftarrow$ the item registered in N ;
- (7) **For** each ancestor of N , N_a , **do**
- (8) $i_x \leftarrow$ the item registered in N_a ;
- (9) **If** $i_x i_y \in F_2$, **then**
- (10) $i_x i_y.support \leftarrow i_x i_y.support + N.account$;
- (11) **Else**
- (12) $i_x i_y.support \leftarrow N.account$;
- (13) $F_2 \leftarrow F_2 \cup \{i_x i_y\}$;
- (14) **Endif**
- (15) **Endfor**
- (16) **For** each itemset, P , in F_2 **do**
- (17) **If** $P.support < \xi \times |DB|$, **then**
- (18) $F_2 \leftarrow F_2 - \{P\}$;
- (19) **Else**
- (20) $P.Nodeset \leftarrow \emptyset$;
- (21) **Endif**
- (22) **Endfor**
- (23) Scan the POC-tree by the pre-order traversal **do**
- (24) $N_d \leftarrow$ currently visiting Node;
- (25) $i_y \leftarrow$ the item registered in N_d ;
- (26) **For** each ancestor of N_d , N_{d_a} , **do**
- (27) $i_x \leftarrow$ the item registered in N_{d_a} ;
- (28) **If** $i_x i_y \in F_2$, **then**
- (29) $i_x i_y.Nodeset \leftarrow i_x i_y.Nodeset \cup N_d.N_info$;
- (30) **Endif**
- (31) **Endfor**
- (32) $F \leftarrow F \cup F_1$;
- (33) **For** each frequent itemset, $i_s i_t$, in F_2 **do**
- (34) Create the root of a tree, R_{st} , and label it by $i_s i_t$;
- (35) **Constructing_Pattern_Tree**($R_{st}, \{i \mid i \in F_1, i > i_s\}, \emptyset$);
- (36) **Endfor**
- (37) **Return** F ;

Algorithm – 4.3: Zero-First Technique

Input: size – k locally large itemsets and broadcasting site list

S_i ($i = 1, 2, \dots, n$)

Output: polling site list for size –k itemset.

1. for all site
2. if candidate set by sites are different size C_k
3. arrange the sites in order of the number of locally large itemset broadcasted S_i
4. remove the sites from list with max. number of locally large itemset broadcasted $LL_{i(k)}$
5. for all locally large itemsets
6. arrange in order removing duplicates $LL_{i(k)}$
7. for all ordered candidates
8. assign the polling site in zero – first order
9. if candidate set exhausted
10. return polling site list for k – candidate set
11. else
12. repeat the same order of sites and continue assignment

Algorithm- 4.4 : QDFIN

Input: Partitioned database DB_i ($i = 1, 2, \dots, n$)

Output: L: set of all globally large itemsets.

Method: Execute the code for all k – itemsets at all sites, starting from k = 1 th size greater than 1

- 1) for all sites
- 2) for k = 1
- 3) find the support count $T_{i(1)}$
- 4) construct the POC tree using POC FIN algo
- 5) for k > 1

- 6) find the size – 2 itemset using FIN algo
- 7) (scan the POC to find size – 2 itemset)
- 8) for all itemsets Z belongs to frequent itemsets $T_{i(k)}$
- 9) if support count is locally large then
- 10) for all nodes
- 11) insert itemset into locally large list $LL_{i(k)}$
- 12) broadcast to all sites
- 13) for all sites
- 14) for all itemsets Z belongs to locally large $LL_{i(k)}$
- 15) insert itemset into local pruning set
- 16) using Zero – first, get the list of polling sites
- 17) for all sites,
- 18) send locally large $LL_{i(k)}$ to polling site S_m
- 19) for all itemsets Z belong to local pruning LP
- 20) send polling request for itemset Z to all sites
- 21) all sites reply polling request from $(T_{i(k)})$
- 22) send support counts $Z. sup_m$
- 23) for all itemsets Z in the polling set $LP_{i(k)}$
- 24) receive support count $Z. sup_m$ from all sites
- 25) for all the itemsets
- 26) calculate global support $Z. sup$ by
sumup of all local support where
- 27) if $Z. Sup >$ the global support threshold
- 28) Add to global frequent itemset $G_{i(k)}$
- 29) broadcast global frequent itemset $G_{i(k)}$;
- 30) if $(k = 1)$ remove_infrequent(DB_i);
- 31) Generate set of all globally large itemsets
- 32) return globally large $k -$ itemset $L_{(k)}$ for generation of frequent itemset for next iteration

The steps in the algorithm are explained below:

- (i) Database DB_i for all partitions are scanned, local counts for all items of size-1 are found and POC-tree is constructed using FIN algorithm[85]. This is responsible to generate candidate set $CG_{i(k)}$ at all sites locally. If $k>1$ in the next pass, it uses POC-tree to find the candidate sets using FIN algorithm. If $CG_{i(k)}$ is empty, no k size itemsets are found then the process stops. (line 1-6)
- (ii) At all the sites, locally large itemsets of size- k are found by local pruning where the count is more than the minimum local support threshold 's' and generate the locally large $LL_{i(k)}$ itemset for broadcast.

The locally large itemset $LL_{i(k)}$ by all the sites are broadcasted to all other sites. This information is used for finding the globally large itemsets. (line 7-15)

- (iii) Zero-first algorithm receives list of sites and locally large items communicated by each site. It returns the list of all polling sites for size- k locally large itemsets $LL_{i(k)}$ which is communicated to all the sites S_i by the coordinating site. (line 16-20)
- (iv) Polling sites store the itemsets in $LP_{i(k)}$ and store the list of sites and itemsets $Z.large_sites$. The polling sites S_i receive these local frequent items $LL_{i(k)}$. All other sites send the support counts of itemsets to the polling sites. (line 21-24)
- (v) At the polling sites after receiving all counts, computes the global counts for locally large items $LL_{i(k)}$. Then the global count of each itemset is compared with the minimum support count condition to find the global large itemsets. The globally large itemsets are stored in $G_{i(k)}$ and broadcasted to all other sites. (line 25-29)
- (vi) A home site receives the frequent itemsets. If it is the first pass, then the dataset is updated. All the infrequent itemset of size-1 are removed from the database. A final set of global large itemsets are returned. To find the 2-itemset, all local large itemsets having size greater than 1, repeat the process step 1. Remove all infrequent items having count less than minimum the globally large size-1 items. The POC-tree is scanned. This generates the 2-itemset and nodeset structure and so on. These local large itemsets are sent to respective polling sites. (line 30-32).

4.1.1 Efficiency of Local Frequent Mining

All the sites use efficient Nodeset data structure[85] to find locally large itemsets and create an efficient POC-tree. All the frequent 1-itemsets are stored in the POC tree. For finding frequent 2-itemsets and nodeset the POC tree is scanned. Then delete the infrequent itemsets and initialize the nodesets of all frequent 2-itemsets by null. Using the preorder traversal, generate the nodesets of all frequent 2-itemsets. Then the same procedure is used to generate the frequent-k itemsets. This reduces the number of database scans and improves the performance. The nodeset is an efficient data structure, this helps in further reducing the scan time. Table 3.1 is the part of the transaction database for size-1 items. Figure 3.3 shows the POC tree construction for the data.

4.1.2 Distributed Database and Resources

Database is distributed across the globe amongst different sites $\{S_1, S_2, \dots, S_n\}$ so called distributed database DB_i . Each site finds the local frequent itemset at each site LL_i . Each site is being used and gives throughput, overall throughput is proportional to the number of nodes or sites. There is a trade-off between the communication load for processing throughput in order to get the best performance out of the setup depending of number of sites. There is not only one centralized site which processes and finds the frequent itemsets but all sites behave as home as well as polling site. Sites have small data portion only and used for finding local FI. It takes less time to scan and create nodeset, also less capabilities and memory to process the small data as compared to sites with large data. These nodes are under-utilized. Zero-first technique takes care of the highly loaded sites and assign polling first to less loaded sites for generating the globally large itemsets. It is a good load balancing technique which utilizes the less loaded sites by effective use

of resources. All the sites are involved in processing the support count and finding the global frequent itemsets for a specific local large itemset hereby reducing the load to find global frequent itemset on one centralized site. In this algorithm all sites participate in the process of the local and then global frequent itemsets and good amount of parallelism is achieved.

4.1.3 Communication Load Reduction

At each site, there is a process of pruning which reduces the size of the candidate sets. Whenever any site finds the candidate sets of frequent itemsets, if it is not locally large to that particular site, that site remove that itemset from the candidate set by the means of local pruning. If frequent size-2 itemset at site -2 {ab, ac, bf, cf}. After pruning itemsets having support count less than the minimum support threshold say {cf, bf} are removed from the candidate sets. The remaining candidate sets {ab, bc} communicated to all other sites. The pruning process reduces the number of candidate sets drastically hence reduce the load on the network and communication cost. After finding the global counts of the candidate sets, global pruning also reduces the load on the communication load. This process makes it network efficient algorithm as small set of data is being communicated to all sites.

4.2 EXPERIMENTAL EVALUATION

This section deals with the environments used to evaluate the performance of the proposed algorithm by comparing it with the popular distributed data mining algorithms FDM [23] and PFIN [19].

4.2.1 Experimental setup

In the experiments, real dataset mushroom is used, which is often used by many researchers in study of frequent itemset mining, for testing the performance of algorithms. The specifications of the mushroom dataset are given in Table 4.1.

Table 4.1 Specifications of Mushroom Dataset

Dataset	Total Trans.	Number of Items	Average Length	Type	File Size
Mushroom	8124	119	23	Dense	570 KB

The dataset mushroom is divided horizontally on 4, 5 and 6 nodes. Two experiments are performed, one on uniform data partition sizes at each site and another on varying data partition sizes, given in Table 4.2 and 4.3.

Experiment-1: Uniform data partitions sizes

Table 4.2 Uniform Data Partition sizes at different sites

No. of Nodes	DB ₁	DB ₂	DB ₃	DB ₄	DB ₅	DB ₆
4	2030	2030	2030	2034		
5	1625	1625	1625	1625	1624	
6	1350	1350	1350	1350	1350	1374

Experiment-2: Varying data partition sizes

Table 4.3 Varying data partition sizes at different sites

No. of Nodes	DB ₁	DB ₂	DB ₃	DB ₄	DB ₅	DB ₆
4	812	1625	2437	3250		
5	406	1219	1625	2031	2843	
6	244	731	1137	1544	2031	2437

4.3 PERFORMANCE ANALYSIS

The proposed algorithm QDFIN is compared with FDM and PFIN(Parallel FIN)[19] algorithms on the basis of execution time. In the proposed algorithm, all sites participate in the processing giving better throughput and pruning technique reduces the number of candidate sets resulting low communication overhead. It assigns the polling site using new presented technique zero-first which further reduces load on the fully occupied sites. At each site an efficient data structure nodeset based on POC-tree[85] is used which reduces number of data scans and hence improve the performance by reducing the data access time.

The algorithms are run on Mushroom dataset with minimum supports thresholds of 10, 20, 30, 40, 50, and 60 percent. Experiments are done on 4, 5, and 6 nodes setups and are compared on the basis of execution time.

The data scan in FDM algorithm is high and its scan the local database at each site in every pass for finding support count for k-itemsets, where $k = 1, 2, \dots, n$. The data scan in PFIN and QDFIN algorithms is only once at the beginning for $k=1$ and they construct POC tree. Once the tree is constructed, these algorithms scan the tree to find higher order itemsets for $k>1$ and saves the time to scan the database partitions stored at various sites again and improves the performance as compared to FDM algorithm.

4.3.1 Comparison on Uniform Data Partition Size

In the first experiment where each site is having uniform data partitions size, QDFIN performs better in all 4, 5, and 6 node setups shown in Figures 4.1 - 4.3 and their execution time are given in Table 4.4 – 4.6.

Table 4.4 Execution time on uniform partition size on 4 nodes (seconds)

Min.Support	QDFIN	FDM	PFIN
10	55.01	67.97	57.03
20	43.05	52.50	45.92
30	35.75	41.95	37.08
40	29.60	33.70	31.11
50	25.45	28.35	26.40
60	22.55	25.65	22.75

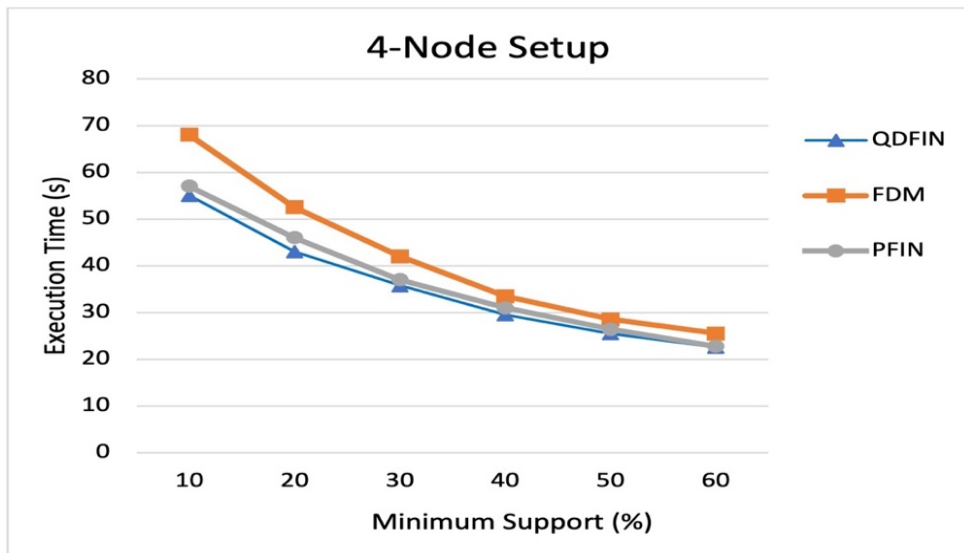


Figure 4.1 Execution time on uniform partition size on 4 nodes

Table 4.5 Execution time on uniform partition size on 5 nodes (seconds)

Min.Support	QDFIN	FDM	PFIN
10	32.11	44.02	37.04
20	25.20	33.35	28.73
30	22.14	27.25	24.35
40	19.07	22.80	21.05
50	16.65	20.19	18.11
60	15.10	18.25	16.50

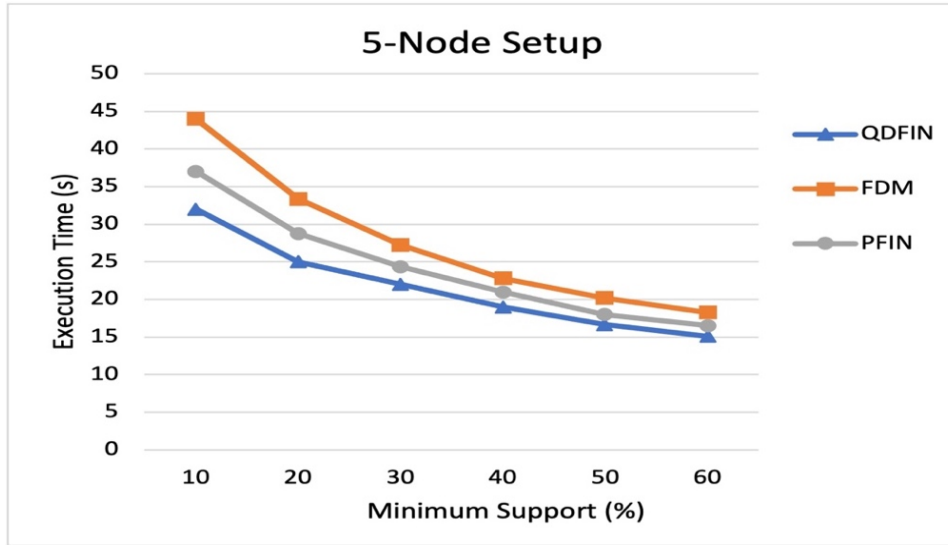


Figure 4.2 Execution time on uniform partition size on 5 nodes

Table 4.6 Execution time on uniform partition size on 6 nodes (seconds)

Min.Support	QDFIN	FDM	PFIN
10	19.18	30.12	24.04
20	16.13	23.23	19.45
30	13.21	19.01	15.90
40	10.89	15.98	13.37
50	9.82	14.50	11.76
60	9.02	13.02	10.60

Figure 4.1 shows in 4-node setup, execution time of QDFIN is close to the execution time of PFIN algorithm as both the algorithm use nodesets data structure for finding locally large itemsets, which reduces the scan time. But it performs better than the FDM algorithm. In the 4-node setup all the sites are producing candidate sets and the advantage of the zero-first technique is very small. The same algorithms are also compared in the 5-nodes and 6-nodes setups. Figures 4.2 and 4.3 show that the proposed algorithm QDFIN performs better in 5 nodes. It further improves in 6-node setup as with the increase

of number of nodes, number of locally large k-itemsets further reduces with the increase of k. It makes some imbalance in the sites in broadcasting the number of candidate sets where some sites become less occupied or free. The zero-first technique assigns the polling site in order of their occupancy for finding globally large itemsets. Figures 4.2 and 4.3 shows the same effect where the performance of the QDFIN is better than the other algorithms.

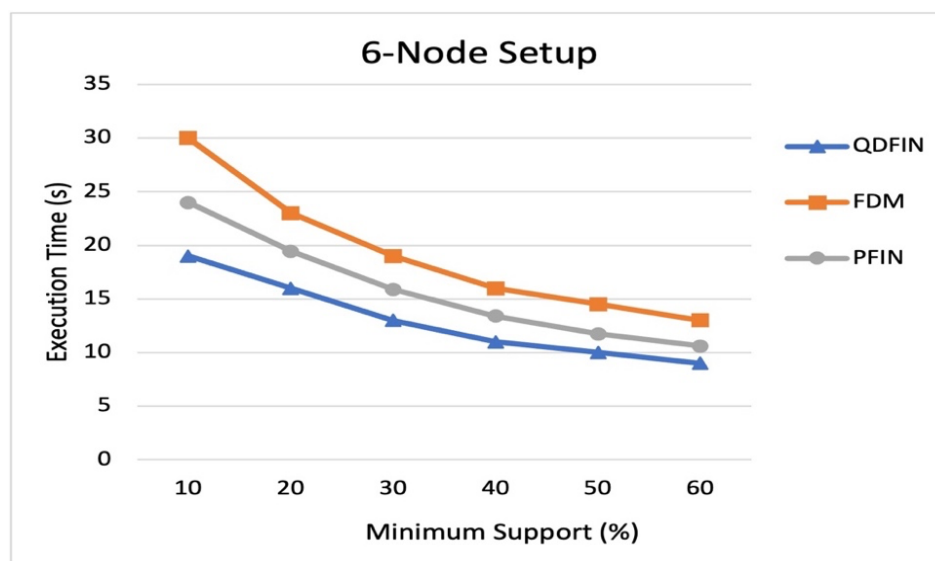


Figure 4.3 Execution time on uniform partition size on 6 nodes

4.3.2 Comparison on Varying Data Partition Size

In the second experiment, same algorithms are compared with varying size of the data partitions on various sites. Tables 4.7-4.9 shows the execution time of all algorithms. Figures 4.4, 4.5, and 4.6 show that the performance of the QDFIN is best in all three setups.

Table 4.7 Execution time on varying partition size on 4 nodes (seconds)

Min.Support	QDFIN	FDM	PFIN
10	41.02	94.13	61.90
20	31.89	65.63	45.48
30	26.48	52.44	37.86
40	21.93	42.13	30.00
50	18.85	35.44	25.14
60	16.70	32.06	21.67

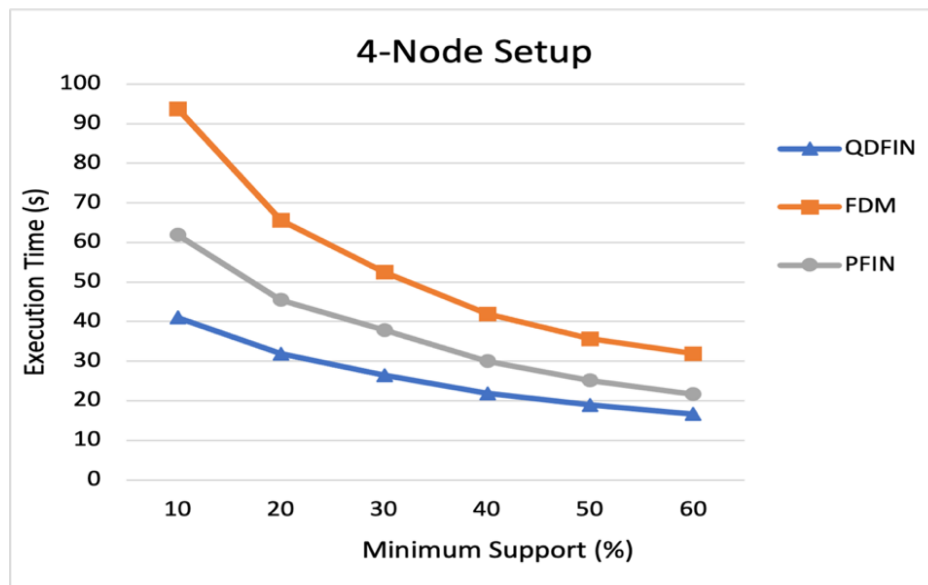


Figure 4.4 Execution time on varying partition size on 4 nodes

Table 4.8 Execution time on varying partition size on 5 nodes (seconds)

Min. Support	QDFIN	FDM	PFIN
10	18.53	52.00	34.77
20	14.12	39.02	26.19
30	12.12	30.96	22.12
40	10.88	26.08	19.14
50	9.79	21.89	16.45
60	8.88	19.00	14.32

The time performance of QDFIN is much better due to the reason of load balancing. As the data size available at all nodes differ, implies that number of transactions differ. The sites with fewer number of transactions produces small candidate sets, takes less time to scan the data, use lesser memory, less processing required. QDFIN takes the advantage of the load differences and assigns less loaded sites first as polling site. Some sites have large number of transactions takes more time and processing capabilities to scan, store the data and generate more candidate sets and are highly busy.

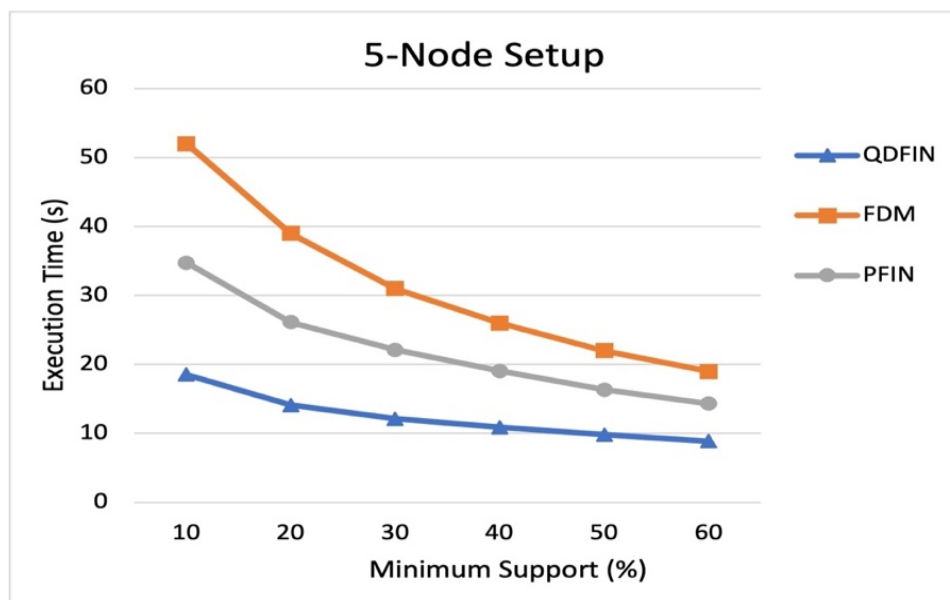


Figure 4.5 Execution time on varying partition size on 5 nodes

The zero-first technique of the proposed algorithm excludes these busy sites from the polling site list. Number of sites increases in Figures 4.5, 4.6 and the difference in number of candidate sets generated by different sites also increase. Some of the sites generate no candidate sets even for $k=1$ or $k=2$, this imbalance further increase in case of 6-node setup as shown in Figure 4.6. It directly effects the load balance, which makes QDFIN best amongst the all three algorithms. The performance difference as compared to

FDM is even more due to the efficient data structure nodeset is used in QDFIN. It is also observed that with low minimum support threshold say 10%, 20%, high number of frequent itemsets are generated and all algorithms take more time and QDFIN performs better due to the difference of candidate set generations amongst each site. The time performance difference reduces for higher minimum support threshold 50%, 60% because smaller number of candidate sets are generated, even some sites generate zero frequent itemsets and QDFIN performs better by load balancing.

Table 4.9 Execution time on varying partition size on 6 nodes (seconds)

Min.Support	QDFIN	FDM	PFIN
10	11.00	40.09	0.26
20	8.37	28.93	19.45
30	6.97	23.11	15.91
40	6.14	17.97	13.40
50	5.51	16.01	11.75
60	4.91	15.06	10.60

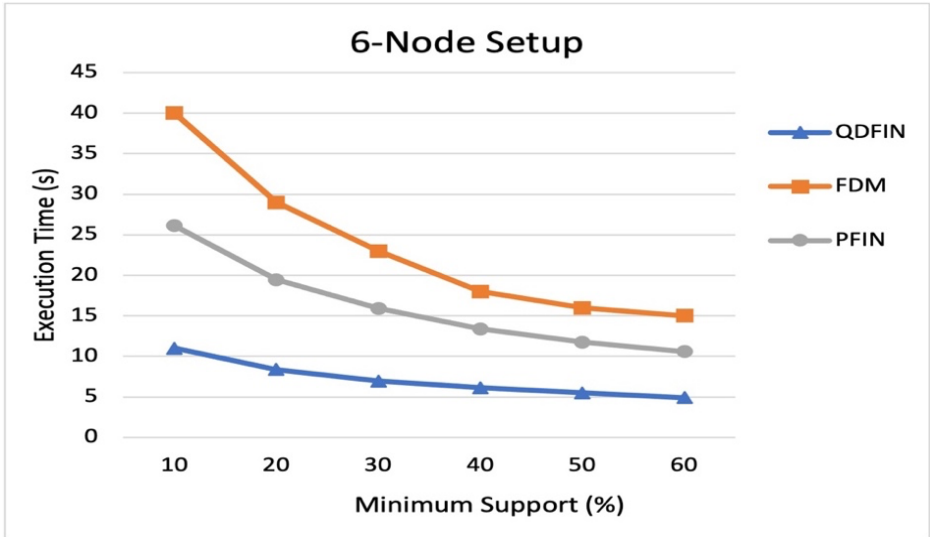


Figure 4.6 Execution time on varying partition size on 6 nodes

QDFIN performs best in lower as well as higher minimum support count in both the experiments i.e. with varying or uniform size data partitions in all three setups 4, 5, and 6 nodes as compared to the PFIN and FDM algorithms. It also shows that with the increase of the number of nodes or decrease in the minimum support threshold QDFIN outperforms other similar algorithms specially in varying data size. The size of data has impact on the execution time[57]. It uses the advantages of the POC-tree and nodeset data structure locally by saving scan time, pruning in reducing communication load and zero-first technique for load balancing.

4.4 SUMMARY

New algorithm QDFIN for varying data partition size for DARM is proposed in this chapter. New efficient data structure nodeset[85] is used in the proposed QDFIN algorithm to generate the frequent itemsets at each site, scan dataset once, reduce communication load by pruning and balance the load on sites by zero-first technique. The algorithm performance evaluation is done on 4-nodes, 5-nodes, and 6-nodes setup with different support threshold. Two experiments are performed, one with uniform and other with non-uniform data partition sizes available on different sites. The performance of the proposed algorithm is compared with similar latest algorithms FDM and PFIN[19]. It outperforms the other algorithms in both the experiments.

5 SIZE BASED DISTRIBUTED ASSOCIATION RULE MINING

This chapter discusses the Size Based Distributed Association Rule Mining (SBDARM) algorithm. It uses a novel technique of size based assignment (SBA) of polling site for finding globally large itemsets based on the data size available at each site. Globally large itemsets are found by sites which are less occupied hence increase the overall computational capabilities and improve the performance. It uses local as well as global pruning to reduce the candidate sets. There is no-broadcasting of candidate sets which further reduces the load on the communication network.

5.1 PROPOSED ALGORITHMS

Symbol Description [23]

s – minimum Support;

D - Total transactions;

L_k – Globally large k -itemsets;

$Z.\text{sup}$ - Global support count of Z ;

CA_k – Candidate sets size k ;

D_i - transactions in partition S_i ;

$GL_{i(k)}$ – globally large itemset size k at S_i ;

$CG_{i(k)}$ - Candidate sets size k at site S_i ;

$LL_{i(k)}$ - Locally large size- k itemsets in $CG_{i(k)}$;

$Z.\text{sup}_i$ – Local support count of Z at S_i

$LP_{i(k)}$ – Local pruning k -itemset at site S_i

Algorithm-5.1: Size Based Assignment Technique (SBA)

Input: Locally large k-itemsets from each site CG_i and data size of all sites $S_i(i = 1,2, \dots n)$

Output: Assigned Polling sites list

1. if $k = 1$ itemsets
2. for all site
3. find the average number of transactions each site in percentage AT
4. Compute the transaction size at each site S_i in percentage PS_i
5. Load difference at site in S_i in percent Δ_i
6. for all locally large itemsets
7. arrange in order removing duplicates $LL_{i(k)}$
8. compute the average candidate sets per site in percentage ACG
9. for all sites
10. for all ordered candidates
11. assign the polling site in size based assignment
average candidates sets plus load difference in percent
12. broadcast the polling site list for $k -$ candidate sets

Algorithm- 5.2 : Size Based Distributed Association Rule Mining (SBDARM)

Input: database DB_i ($i = 1,2, \dots n$)

Output: Globally large itemsets.

Method: Running algorithm for all k-itemsets for $k=1..n$, on all partitions

1. for all sites
2. for $k = 1$
3. find the support count $T_{i(1)}$
4. construct the POC tree using POC FIN algorithm
5. for $k > 1$
6. find the size - 2 itemset using FIN algo
7. (scan the POC to find size - 2 itemset)
8. for all itemsets Z belongs to $k -$ itemsets
9. $T_{i(k)}$ generate local pruning list

10. if support count is locally large then
11. for all sites
12. insert itemset into locally large list $LL_{i(k)}$ for all sites
13. communicate locally large list $LL_{i(k)}$ and data size
to coordinating site for assignment of polling site
14. for coordinating site call SBA get list of polling sites
15. for all sites,
16. send locally large $LL_{i(k)}$ to polling site S_m
17. for all itemsets Z belong to local pruning LP
18. send polling request for itemset Z to all sites
19. all sites reply polling request from $(T_{i(k)})$
20. send support counts $Z.\text{sup}_m$
21. for all itemsets Z in the polling set $LP_{i(k)}$
22. receive support count $Z.\text{sup}_m$ from all sites
23. for all the itemsets
24. calculate global support $Z.\text{sup}$ by sumup of all local support where
25. if $Z.\text{Sup} >$ the global support threshold
26. Add to global frequent itemset $G_{i(k)}$
27. broadcast global frequent itemset $G_{i(k)}$;
28. if $(k = 1)$ remove_infrequent(DB_i);
29. Generate set of all globally large itemsets
30. return globally large $k -$ itemset $L_{(k)}$

The steps are discussed below:

- (i) Database DB_i at all sites are scanned, local size 1 itemsets are found and POC tree is created. For $k > 1$ local itemsets are found using FIN and nodesets by reverse scan of POC tree. This generates locally itemsets and nodeset structure from all partitions (line 1-7)

- (ii) Local pruning is done to generate candidate sets, locally large k-itemsets $LL_{i(k)}$ having the count greater than the minimum support threshold. (line 8-12)
- (iii) The locally large itemset $LL_{i(k)}$ are communicated to the site which assigns the polling sites to the locally large itemsets. Size based assignment algorithm receives list of sites with number of transactions and locally frequent items communicated by each node. SBT communicates the list of polling sites for locally large k-sized itemsets $LL_{i(k)}$ sites S_i .(line 13-14, call algorithm 1)
- (iv) All the sites S_i send the local counts for the locally large items $LL_{i(k)}$ to the polling sites assigned in the last step. Polling sites store all information about the itemsets in $LP_{i(k)}$ and $Z.large_sites$. (line 15-20)
- (v) Each Polling site receives counts, computes global counts for assigned locally large itemsets $LL_{i(k)}$. It generates the global large itemset, stores in $G_{i(k)}$ after removing the itemsets having counts less than the support threshold value. Then globally large itemsets are communicated to all sites. (line 21-27)
- (vi) All home site receives the global frequent itemsets, update and remove all infrequent 1-itemset. In the next pass home sites find the 2-itemset, i.e., locally large size k ($k=2\dots n$), repeat the process. Remove all infrequent k-itemsets. (line 28-30)

5.1.1 Efficiency at Each Site

There are n number of sites $\{S_1, S_2, \dots, S_n\}$ where database is partitioned and stored, called distributed database DB_i . Sites generate the local frequent k-itemset ($k=1\dots n$) using efficient algorithm. Polling sites are assigned on the basis of the data partition size, where site with less number of transactions are assigned more local frequent itemsets for finding

global frequent itemsets and sites handling bigger partition size are assigned less workload considering the number of transactions at each site. Sites with small data sizes use less memory, capabilities etc in handling the data, so less occupied and the same is taken care by the proposed algorithm. Size based assignment technique is developed which considers the load on each site while assigning the polling sites for finding the globally large itemsets. It allocates less load to the sites have large data partition and more load to less occupied sites with small data partition. The proposed technique is best for the unbalanced data partitions for the effective use of the resources and balancing the load for finding the globally large itemset using locally large itemset. This technique utilises all resources and as all the sites participate as per their availability so there is no extra load on a centralized or coordinating site and any other site in unbalanced way. This ensures a good amount of parallelism in the real distributed database where centralized database has no control on the partitions.

5.1.2 Low Communication Overhead

The algorithm also takes care of the load on the communication channel by reducing the size of the candidate sets by pruning at each site. All the sites first find the frequent itemsets and then through pruning process remove non frequent itemsets having counts less than the required support counts to become eligible for communication and may not be globally frequent.

Let frequent 2-itemset at site 3 be {ad, eg, jg, ht }. After applying pruning process, removing not eligible itemsets where support count is less than the minimum support threshold i.e. {ad, jg} are removed. The reduced set {eg, ht } after pruning is communicated to the site for the assignment of polling sites. The technique reduces the

size of the candidates sets to half and reduces the communication overhead. The algorithm uses no-broadcasting technique where all sites send candidate sets to the coordinating site for polling site assignment, in place of broadcasting it to all sites. It reduces the number of candidate set communication to $O(n)$ messages and hence there is less load on the communication network. This technique is network efficient with reduced size of the data communicated.

5.2 EXPERIMENTAL EVALUATION

This section deals with the environments used to evaluate the performance of the proposed algorithm SBDARM by comparing it with the distributed data mining algorithms FDM [23] and PFIN [19].

5.2.1 Experimental setup

Functionality and performance of the algorithms are tested on four datasets including three real datasets mushroom, connect, chess and one synthetic dataset T10I4D100K. Datasets specifications are given in Table 5.1.

Table 5.1 Specifications of datasets used

Dataset	Total Trans.	Number of Items	Average Length	Type	File Size
Mushroom	8124	119	23	Dense	570 KB
Connect	67557	129	43	Dense	9.3 MB
Chess	3196	75	37	Dense	342 KB
T10I4D100K	1,00,000	1000	40	Sparse	4 MB

Experiments are performed on a cluster of five nodes connected with LAN, using Intel i5 64 bit processor running @ 3.3 GHz with 6 GB RAM, 1 TB HDD, running on windows 10 OS, having Java JDK 1.7. Database is partitioned and stored at five nodes with varying size. Data partitions distributed to each site are based on number of transactions, shown in Table 5.2.

Table 5.2 Details of the data partitions available at different sites for different datasets

	Partition Size	Mushroom	Connect	Chess	T10I4D100K
DB ₁	10%	812	6756	320	10000
DB ₂	15%	1219	10134	479	15000
DB ₃	20%	1625	13511	639	20000
DB ₄	25%	2031	16889	799	25000
DB ₅	30%	2437	20267	959	30000
Total	100%	8124	67557	3196	100000

5.3 PERFORMANCE ANALYSIS

The new proposed algorithm SBDARM implemented and executed for execution time performance comparison with some of the existing algorithms FDM and PFIN. In the experiments, algorithms are compared on execution time where sites have varying data sizes of the data partitions shown in Table 5.2. In the first experiment all algorithms are executed on dataset mushroom with varying minimum support threshold values i.e. 10%, 20%, 30%, 40%, 50%, and 60%. Table 5.3 shows the local frequent 1-itemsets generated at each site on the mushroom datasets. Similarly in the second experiment these are run on connect dataset with same minimum support threshold 10%, to 60%. Third experiment is performed on chess dataset with 10%, 15%, 20%, 25%, 30%, 35% minimum support

threshold. Lastly on T10I4D100K dataset with minimum support threshold 0.1%, 0.2%, 0.3%, 0.4%, 0.5%, and 0.6%. The local frequent 1-temsets generated by the proposed algorithm at each site on connect, chess and T10I4D100K datasets are shown in Table 5.4, 5.5 and 5.6 respectively.

The SBDARM algorithm effectively uses the resources at all the sites, reduces the local load and communication load. It gives best throughput by using pruning techniques for reducing candidate sets for communication along with the size based assignment technique. There is no-broadcasting of frequent itemsets by all the sites rather all sites communicate the frequent itemsets to one coordinating site for the assignment of the polling site. The frequent itemset message exchange in proposed algorithm by n nodes to one coordinating node $n \times 1$ i.e. $O(n)$ whereas in other algorithms its n nodes sending to n nodes, so $n \times n$ which is $O(n^2)$. This reduces the number of messages and hence load on the communication network. The number of data scans performed by the proposed algorithm and PFIN algorithm is only once whereas database is scanned in every pass in FDM hence effect the performance due to delay in I/O operations.

5.3.1 Generating Local Itemsets

Table 5.3 Local frequent 1-itemsets generated at each partition on Mushroom dataset

Partition	----- Support Count Threshold (%) -----					
	10	20	30	40	50	60
DB1	44	34	26	19	16	12
DB2	48	37	30	25	18	14
DB3	39	37	26	23	17	15
DB4	51	38	27	20	18	13
DB5	50	41	31	22	19	14

Table 5.4 Local frequent 1-itemsets generated at each partition on Connect dataset

Partition	----- Support Count Threshold (%) -----					
	10	20	30	40	50	60
DB1	66	54	46	40	35	33
DB2	69	56	46	43	38	35
DB3	69	57	47	42	40	38
DB4	68	56	49	42	37	37
DB5	70	56	46	43	40	36

Table 5.5 Local frequent 1-itemsets generated at each partition on Chess dataset

Partition	----- Support Count Threshold (%) -----					
	10	15	20	25	30	35
DB1	52	50	47	46	44	42
DB2	52	50	48	46	42	41
DB3	55	53	48	46	44	41
DB4	61	58	58	53	50	43
DB5	66	57	54	52	51	49

Table 5.6 Local frequent 1-itemsets generated at each partition on T10I4D100K dataset

Partition	----- Support Count Threshold (%) -----					
	0.1	0.2	0.3	0.4	0.5	0.6
DB1	789	736	682	621	559	513
DB2	793	742	684	630	559	516
DB3	798	740	689	633	563	520
DB4	794	738	691	625	566	515
DB5	794	743	690	628	562	519

The frequent itemsets are sent to one coordinating site only using no-broadcast technique for the assignment of the polling sites. In the proposed algorithm the polling site

assignment is done using size based assignment technique.

5.3.2 Polling sites assignment

Table 5.7-5.10 show that the local frequent 1-itemsets assignment of polling site for finding the global frequent itemsets by the SBDARM algorithm. The same process is repeated for k-itemsets for all $k > 1$. This assignments balance the load on the sites as it allocates the load for finding the global frequent itemsets i.e. assignment of polling sites inversely proportional to the partition sizes on sites. In the other two algorithms the assignments are not based on the size of the partition rather using some hash function or count distribution. The assignment in FDM and PFIN increases the load on the already occupied sites and load balancing is poor. This size based assignment technique is effective in the distributed data environment with varied data size and it reduces the overall time of execution and improves the performance.

Table 5.7 Pruning sites assignment by SBDARM to local frequent 1-itemsets on Mushroom dataset

Partition	---- Support Count Threshold (%) -----					
	10	20	30	40	50	60
DB1	18	14	10	9	6	6
DB2	15	12	8	7	5	5
DB3	12	9	7	6	4	4
DB4	9	7	5	5	3	3
DB5	5	3	2	0	2	0
Total	59	45	32	27	20	17
Globally Large	56	43	28	21	13	8

Table 5.8 Pruning sites assignment by SBDARM to local frequent 1-itemsets on
Connect dataset

Partition	----- Support Count Threshold (%) -----					
	10	20	30	40	50	60
DB1	23	19	15	14	14	13
DB2	19	16	13	12	11	11
DB3	15	13	10	9	9	9
DB4	12	10	8	7	7	7
DB5	6	5	4	3	3	1
Total	75	63	50	45	44	41
Globally Large	73	59	46	41	38	36

Table 5.9 Pruning sites assignment by SBDARM to local frequent 1-itemsets on
Chess dataset

Partition	----- Support Count Threshold (%) -----					
	10	15	20	25	30	35
DB1	21	19	18	17	17	17
DB2	17	16	15	14	14	14
DB3	14	13	12	12	12	11
DB4	11	10	9	9	9	9
DB5	5	4	6	4	4	3
Total	68	62	60	56	56	54
Globally Large	61	57	54	51	50	45

Table 5.10 Pruning sites assignment by SBDARM to local frequent 1-itemsets on T10I4D100K dataset

Partition	----- Support Count Threshold (%) -----					
	0.1	0.2	0.3	0.4	0.5	0.6
DB1	240	223	209	189	172	157
DB2	200	186	174	158	143	131
DB3	160	149	140	126	115	105
DB4	120	112	105	95	86	79
DB5	80	73	68	62	55	50
Total	800	743	696	630	571	522
Globally Large	796	740	691	628	568	516

5.3.3 Discussion

Figures 5.1 – 5.4 show that the performance of the proposed algorithm on all four datasets outperforms the other two algorithms in time execution. The time performance of SBDARM is best due to the load balancing amongst the sites. Number of transactions at each site differ means the resource utilization also differs. The sites with more number of transactions takes more time for data scan, use more memory, and more processing and pruning time. SBDARM utilises the load differences as edge over other algorithms by assigning more load to less loaded sites as compared to highly loaded sites. Some sites with less number of transactions, are having less load of processing, data scan, generate less number of candidate sets and are comparatively less occupied. This is the best load balancing at each site by the assignment of polling sites for finding global frequent itemset inversely proportional to the data partition size available at each site. The number of local frequent k -itemsets for $k > 1$, are further reduced and the SBA algorithm further balance the load amongst the sites and gets performs better.

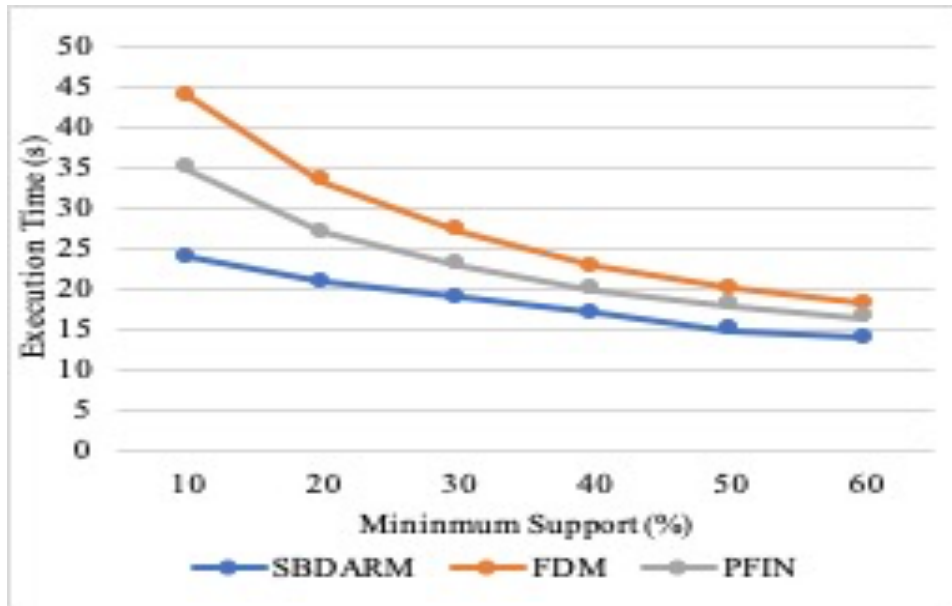


Figure 5.1 Execution time on Mushroom dataset

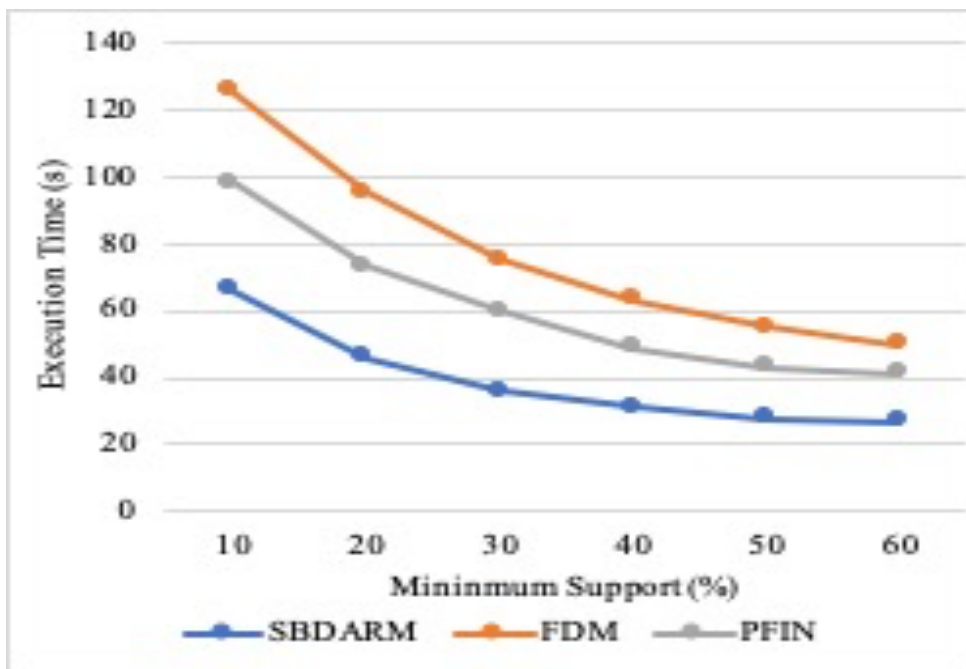


Figure 5.2 Execution time on Connect dataset

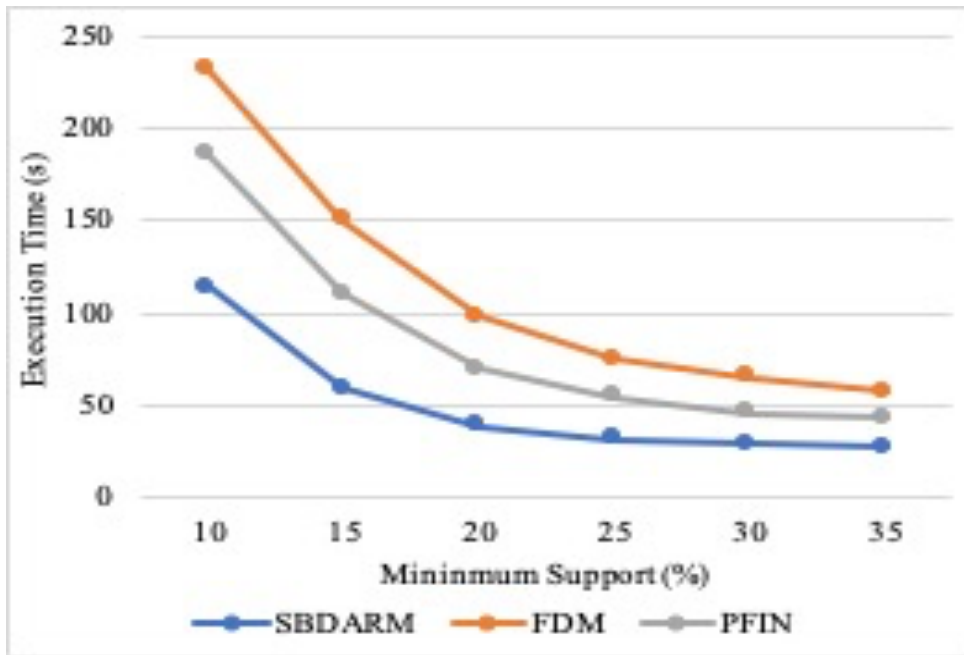


Figure 5.3 Execution time on Chess dataset

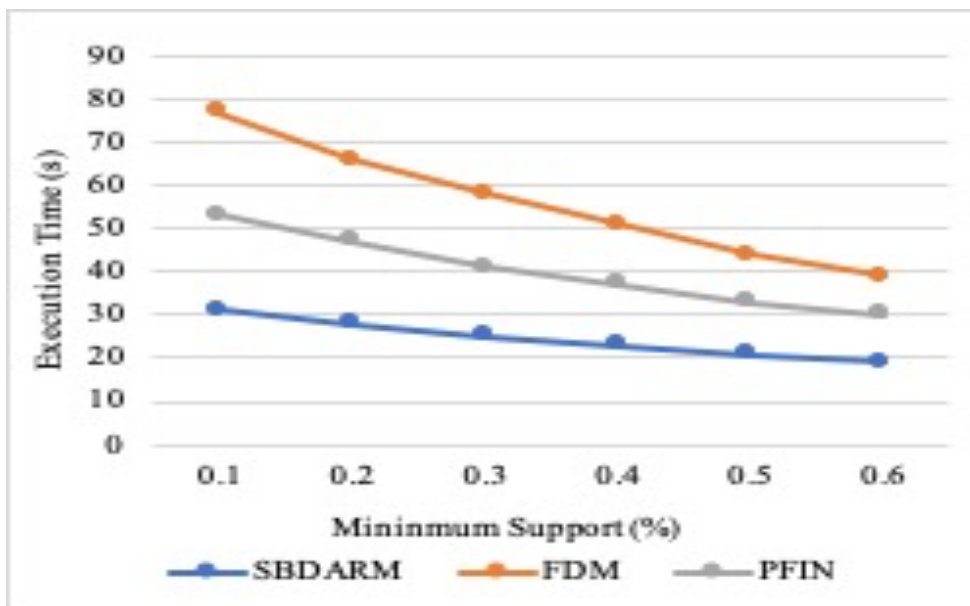


Figure 5.4 Execution time on T10I4D100K dataset

With the increase of the support threshold less number of frequent itemsets are generated and execution time of all the algorithms also reduce. The performance of the proposed algorithm takes less time, although the time difference reduces as number of global frequent itemsets also reduce. The difference in the performance is also due to the reduced communication load, local pruning and no-broadcasting technique which reduces the load on the network and reduces the time delay. In Figure 5.4 dataset used is sparse so when value of k increases to $2,3,\dots,n$ the number of frequent itemsets generated reduce, therefore for low minimum support threshold, the execution time is not very high as compared to the execution time for the high minimum support threshold for all the algorithms.

The analysis shows that the proposed algorithm SBDARM performs best in low and high minimum support threshold in all four comparisons with PFIN and FDM having varying partition sizes. It shows SBDARM outperforms other two algorithms with low minimum support threshold. It uses the advantages of the no-broadcasting by reducing communication and size based assignment reducing load on heavy loaded sites and pruning reducing candidate sets.

5.4 COMPARISON OF SBDARM AND QDFIN

Both the proposed algorithms are compared on two datasets Chess and T10I4D100K. The assignment made by both use different techniques, QDFIN is based on the number of candidate sets generated whereas SBDARM is based on data partition size at each site and SBDARM also uses no broadcasting technique.

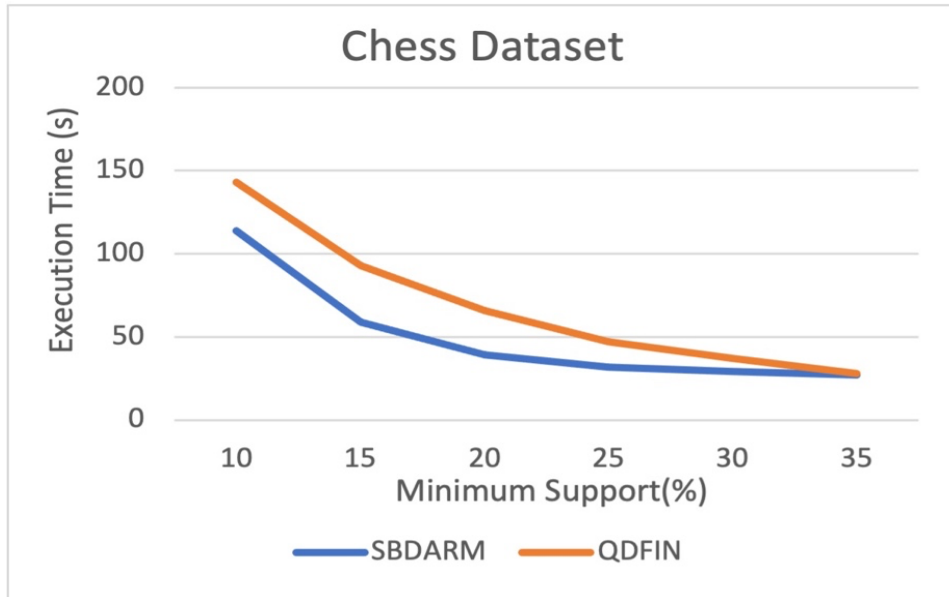


Figure 5.5 Comparison of SBDARM and QDFIN on Chess dataset

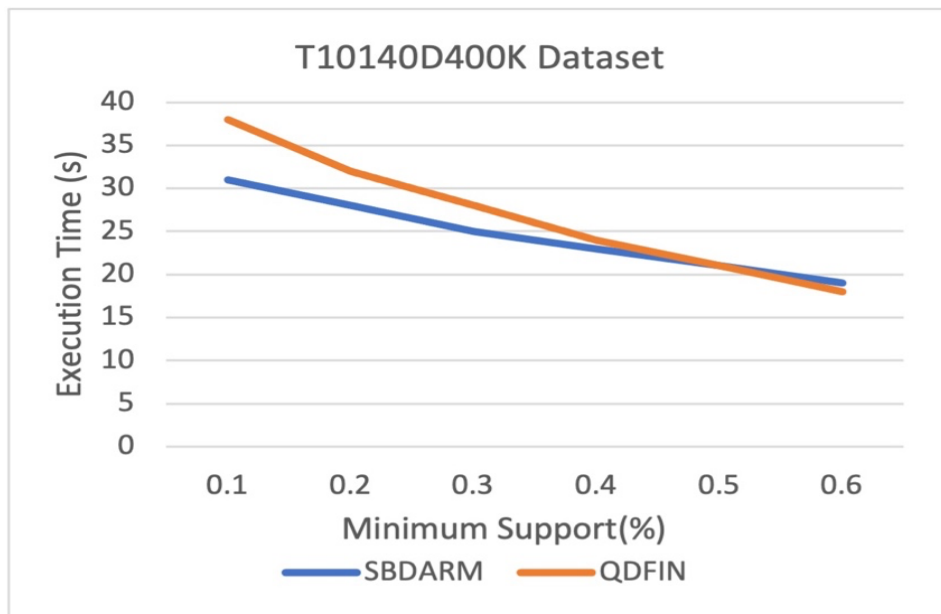


Figure 5.6 Comparison of SBDARM and QDFIN on T10I40D400K dataset

5.4.1 Discussion

SBDARM performs slightly better than QDFIN. In QDFIN the difference in polling site assignment is based on FI generated .i.e.. size of the candidate set, at each site. The polling assignment difference between the sites is just one less for the less loaded

sites. If the size of the candidate set is equal to the highest frequent itemsets generated by any site then that site is not assigned any work. Sites with zero assignment may be more than one, which increase load on other sites. Communication overhead is high in QDFIN if FI are high. Whereas SBDARM allocates load based on the data size available and better load balancing. Communication overhead differs in both the algorithms. In QDFIN all sites broadcast FI to all other site whereas in SBDARM it is communicated to the coordinating site only using no-broadcasting technique. In the sparse datasets, with high support count, a small number of candidate sets are generated and low communication overhead in both the algorithms.

5.5 SUMMARY

This chapter discusses the proposed algorithm SBDARM. The algorithm steps are explained in detail. It is then implemented on four datasets and the results are compared for different support count thresholds and discussed. The proposed algorithm performed best amongst them due to low communication, load balancing and efficient data structure. The algorithm SDDARM is then compared with the other proposed algorithm QDFIN and found slightly better than QDFIN.

6 APPLICATION MODEL – A CASE STUDY

DARM has many commercial applications and useful for the industry functioning in different domains. Many application areas of mining are still not explored well like Tourism industry. Tour and Travel companies would generally have separate branches and data is generated at different branches and applying mining may help the travel companies to improve their business. This chapter explains the application model developed for the tour and travel company by finding the interesting association between different parameters on the basis of the data generated by one such company. It analyses the results and generates rules which may help the industry to use mining for promotion and expanding of their business.

6.1 METHODOLOGY

6.1.1 Distributed Mining of Association Rules

Distributed association rule mining is mainly finding the globally large itemsets from the distributed data for finding association rules. In the distributed data mining, data D_{si} ($i=1,2,\dots,n$) are created or gathered or generated at different sites S_i ($i=1,2,\dots,n$). Let's assume, a virtual dataset $DS = \cup D_{si}$, $|D_{si}|$ is the number of transactions in D_{si} . $|DS|$ is the number of transactions in dataset, for any item A , $A.\text{sup}$ is the support count of A in DS , and $A.\text{sup}_i$ is the support count of A in D_{si} . Given an itemset A , if $A.\text{sup} \geq \text{min sup} \times |DS|$, A is said to be globally frequent itemset; If $A.\text{sup}_i \geq \text{min sup} \times |D_{si}|$, where A is locally frequent itemset on site S . After finding the local frequent itemsets in DARM, global frequent itemsets are generated by merging them and then to find the confidence

which generates the association rules.

6.1.2 Assumptions

- Database is not centralized rather partitioned and distributed amongst sites
- Data is created, gathered or captured at different sites or locations of the company
- Data sizes at various sites vary in size and number of transactions
- There is no centralized database and it is not required or feasible to store data at one centralized location
- There is a coordinating unit having one or more processing units
- All sites participate in the mining process to improve the performance

6.1.3 Algorithm

1. At each site database is scanned to find each combination of parameters age group and destination called itemset.
2. At each site, support count is calculated for each found combination in step 1.
3. Pruning of the itemsets are done by removing the itemsets having support count less than the threshold support count.
4. The remaining itemsets called candidate sets, are communicated to the coordinating site for the assignment of polling sites for finding the global frequent itemsets from all candidate itemsets.
5. Coordinating site assigns the polling sites to all local frequent itemset and communicates to all sites
6. On receiving the polling site details, all sites communicate the local count for the itemset to the polling sites
7. Polling sites calculate the global support count by aggregating the local support counts received from all sites for the assigned itemsets.
8. At the Polling sites, the itemsets having support count less than the global support threshold are removed from the list of frequent itemsets to find the final global frequent itemsets, called global pruning.

9. Polling site then calculate the confidence for the itemsets, perform global pruning by removing the itemsets having confidence less than the minimum confidence thresholds.
10. Association rules are created from the global frequent itemsets generated in the previous step.

6.2 IMPLEMENTATION AND RESULTS ANALYSIS

This research considers a Tour and Travel company “Voyagers Beat” for case study and implementation of the DARM techniques. The tour and travel companies can be classified as small, medium and large, based on different parameters given in Table 6.1. Voyagers Beat is a medium sized organization for the study with booking office at multiple locations and multiple designations within the country having a few lakhs of transactions. The Company has head office in Delhi and has booking offices in 13 cities for organizing package tours for various destinations in India. For the analysis purpose 3 booking sites or distributed database locations are considered. A few destinations are chosen and are grouped into three broad destinations. For experiments two years, 2017 and 2018 booking datasets are considered. Data tables in given in Table 6.2 are taken for analysis.

Table 6.1 Tour and travel companies’ classification

Location	Destinations	Database type	Yearly -number of transactions
Single	single	centralized	Few thousand
single	multiple	centralized	Few thousand
multiple	single	centralized	Few thousand
multiple	multiple	centralized	Few thousand
multiple	multiple	distributed	Few thousand to lakhs
multiple	Inter-national	distributed	Few lakhs
Inter-national	Inter-national	distributed	Lakhs

Table 6.2 Data tables used - Booking-Master and Tourists-Details

Booking Master Table	Tourists_Details Table
Booking_ID	Booking_ID
Date	Name
Booking_Name	Age
Address	
Phone_no	
Destination	
Date_from	
Date_to	
Tot_Amount	
Advance_received	
Mode_Payment	

Attributes Used: Above tables are combined and three attributes are taken for the analysis from the above tables. Final attributes taken for analysis from three data sites are given in the Table 6.3.

Table 6.3 Data attributes taken for analysis

Attributes	Values	Descriptions
ID	Integer	Booking-ID
Age	A1(18-30), A2(31-45), A3(>45)	Tourist-age
Destination	D1(Himachal), D2(Rajasthan), D3(UP)	Place-visited

ID. Identification of the tourist, who is travelling. This attribute is used for reference only and not participating in the mining process.

Age. This attribute is the age of the tourist. This attribute is considered for the analysis by grouping age into three categories i.e. 18 years to 30 years named as A1 group, 31 years to 45 years named as A2 and age greater than 45 is a third group A3.

Destination. There are many destinations where the company operates. Three broad destinations are considered Himachal which includes Kasol, Manali etc., Rajasthan includes Jaipur, Pushkar, Jodhpur etc. and third destination UP by grouping Kanpur, Agra etc.

Some of the destinations are preferred by youngsters and some by elderly people. The study is an attempt to find the association between these two attributes age and destination. Accordingly, the dataset is taken for the analysis and the sample data before and after transformation are shown in the following Tables 6.4.

Table 6.4 (a) Sample data table

ID	Age	Destination
1	24	Kasol
2	54	Agra
3	25	Pushkar
4	48	Kanpur
5	22	Manali
6	35	Agra
7	29	Shimla
8	38	Manali
9	47	Jaipur

(b) Sample data table after transformation

ID	Age	Destination
1	A1	D1
2	A3	D3
3	A1	D2
4	A3	D3
5	A1	D1
6	A2	D3
7	A1	D1
8	A2	D1
9	A3	D2

For the analysis, data from the three sites or locations are chosen. The number of transactions at each site and the total transactions in the database for this research is given in the Table 6.5. Data mining technique is applied independently at each site and the local frequent 1-itemset are found.

Table 6.5 Datasets sizes available at various sites

	Site 1	Site 2	Site 3	Total
Total Transactions	20872	14273	18370	53515

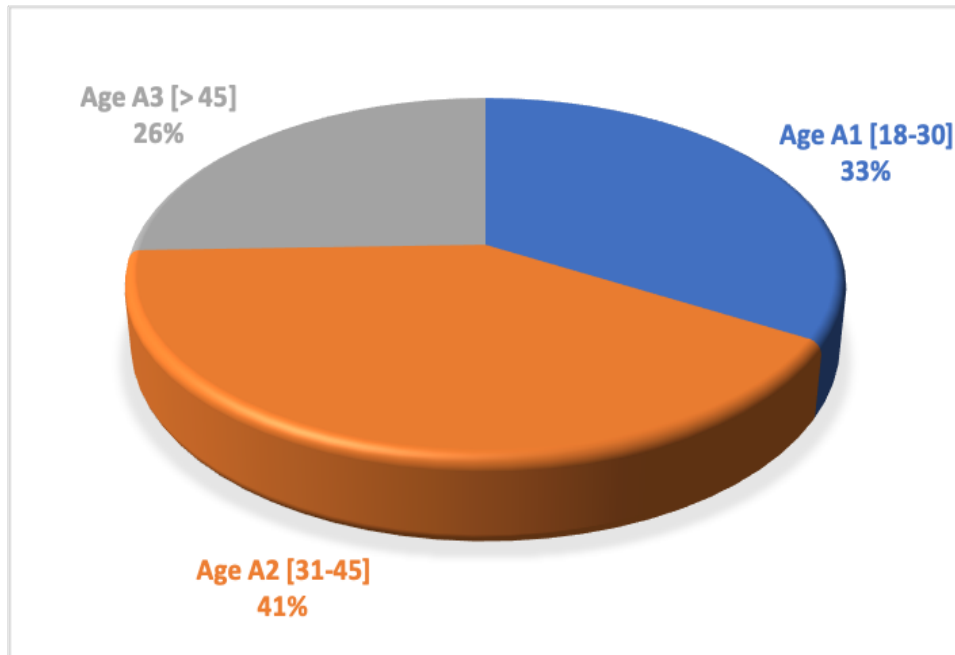


Figure 6.1 Age groups and their support count

The support of each age group and destination is calculated. Figure 6.1 shows the dataset having transaction with age group A2 is 41%, A1 is 33% and A3 is 26% only. Whereas Most preferred destination is D1 at 53%, then D2 at 24% and D3 at 23% shown in Figure 6.2 below. D2 and D3 are almost equally preferred whereas D1 is most visited destination. Then the 2-itemset with the combination of age groups and destination groups and their local support counts at all sites are found see Table 6.6.

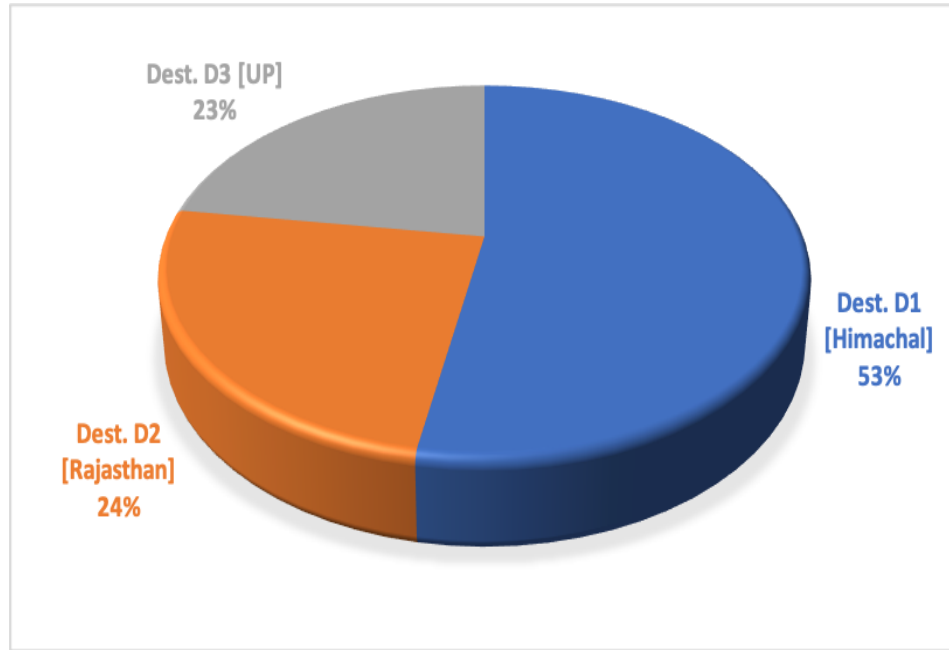


Figure 6.2 Destinations frequency -support count

Table 6.6 Local support count (%) at each site

	Site 1 Support	Site 2 Support	Site 3 Support
Age A1 - Dest. D1	0.261	0.198	0.159
Age A1 - Dest. D2	0.068	0.072	0.106
Age A1 - Dest. D3	0.030	0.014	0.016
Age A2 - Dest. D1	0.281	0.221	0.150
Age A2 - Dest. D2	0.078	0.055	0.094
Age A2 - Dest. D3	0.042	0.029	0.081
Age A3 - Dest. D1	0.025	0.073	0.045
Age A3 - Dest. D2	0.044	0.063	0.078
Age A3 - Dest. D3	0.127	0.098	0.158

To test the model and develop the association rules, two experiments are conducted with minimum threshold support 0.200 and 0.100 given in Table 6.7.

Table 6.7 Support and Confidence threshold values

	Support	Confidence
Experiment 1	0.200	0.500 & 0.300
Experiment 2	0.100	0.500 & 0.300

6.2.1 Experiment 1: Minimum support threshold = 0.200

Local pruning- In this experiment local frequent itemsets are calculated. The itemsets whose support counts is more than 0.200 at least one of the sites, are the candidate sets and are communicated for finding the global frequent itemsets given in Table 6.8.

Global Pruning- The itemsets having global support count less than 0.200 are eliminated. The global support and confidence are calculated for the global frequent itemsets, given in Table 6.9.

Table 6.8 Candidate sets for support count threshold = 0.200

	Site 1	Site 2	Site 3
	Support	Support	Support
Age A1 - Dest. D1	0.261		
Age A2 - Dest. D1	0.281	0.221	

Table 6.9 Global support and confidence

	Global Support	Global Confidence
Age A1 -> Dest. D1	0.209	0.625
Age A2 -> Dest. D1	0.220	0.536
Dest. D1 -> Age A1	0.209	0.441
Dest. D1 -> Age A2	0.220	0.464

(i) Association rules for confidence threshold 0.500

Table 6.10 Association rules for support threshold 0.200 and confidence threshold 0.500

	Global Support	Global Confidence
Age A1 -> Dest. D1	0.209	0.625
Age A2 -> Dest. D1	0.220	0.536

These rules in Table 6.10 show that the destination D1 (Himachal) is the preferred destination for age groups A1 (18-30) and A2(31-45).

(ii) Association rules for confidence threshold 0.300

The rules in Table 6.11 show that the destination D1 (Himachal) is the preferred destination for age groups A1 (18-30) and A2(31-45). Secondly The destination D1 (Himachal) is mostly visited by the young tourists and not by elderly tourists.

Table 6.11 Association rules for support threshold 0.200 and confidence threshold 0.300

	Global Support	Global Confidence
Age A1 -> Dest. D1	0.209	0.625
Age A2 -> Dest. D1	0.220	0.536
Dest. D1 -> Age A1	0.209	0.441
Dest. D1 -> Age A2	0.220	0.464

6.2.2 Experiment 2: Minimum support threshold = 0.100

Local pruning- In this experiment local frequent itemsets are calculated. The itemsets whose support counts is more than 0.200 at least one of the sites, are the candidate sets and are communicated for finding the global frequent itemsets given in Table 6.12.

Global Pruning- The itemsets having global support count less than 0.100 are eliminated. The global support and confidence are calculated for the global frequent itemsets, given in Table 6.13.

Table 6.12 Candidate sets for min. support threshold = 0.100

	Site 1	Site 2	Site 3
	Support	Support	Support
Age A1 - Dest. D1	0.261	0.198	0.159
Age A1 - Dest. D2			0.106
Age A2 - Dest. D1	0.281	0.221	0.150
Age A3 - Dest. D3	0.127		0.158

Table 6.13 Global support and confidence

	Global	Global
	Support	Confidence
Age A1 -> Dest. D1	0.209	0.625
Age A2 -> Dest. D1	0.220	0.536
Age A3 -> Dest. D3	0.130	0.510
Dest. D1 -> Age A1	0.209	0.441
Dest. D1 -> Age A2	0.220	0.464
Dest. D3 -> Age A3	0.130	0.641

(i) Association rules for confidence threshold 0.500

Table 6.14 Association rules for support threshold 0.100 and confidence threshold 0.500

	Global Support	Global Confidence
Age A1 -> Dest. D1	0.209	0.625
Age A2 -> Dest. D1	0.220	0.536
Age A3 -> Dest. D3	0.130	0.510
Dest. D3 -> Age A3	0.130	0.641

These rules show that the destination D1 (Himachal) is a preferred destination for age groups A1 (18-30) and A2(31-45). Also, age A3 group prefers to go to destination D3, and D3 is mostly visited by A3 shown in Table 6.14.

(ii) Association rules for confidence threshold 0.300

Table 6.15 Association rules for support threshold 0.100 and confidence threshold 0.300

	Global Support	Global Confidence
Age A1 -> Dest. D1	0.209	0.625
Age A2 -> Dest. D1	0.220	0.536
Age A3 -> Dest. D3	0.130	0.510
Dest. D1 -> Age A1	0.209	0.441
Dest. D1 -> Age A2	0.220	0.464
Dest. D3 -> Age A3	0.130	0.641

These rules in Table 6.15 clearly indicate that there is a strong association between age A1, A2 and destination D1 where confidence is high implies tourists up to middle age

like to visit destination Himachal, and Himachal is mostly visited by youngsters and middle aged tourists. On the other side A3 age group preference is UP and vice versa as there is a good association between age A3 and destination D3.

6.2.3 Relationships Between Local and Global Rules

The relationship between the local support, global support and confidence are also shown in Figure 6.3-6.5 and discussed below. It also shows some interesting relationships amongst them.

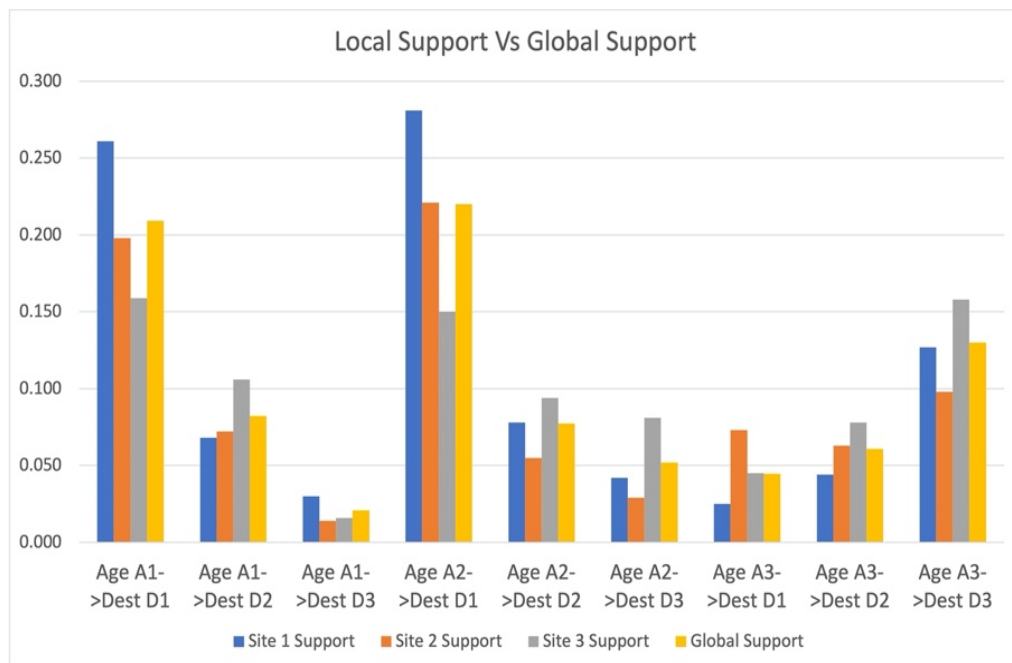


Figure 6.3 Local support count at various sites and global support

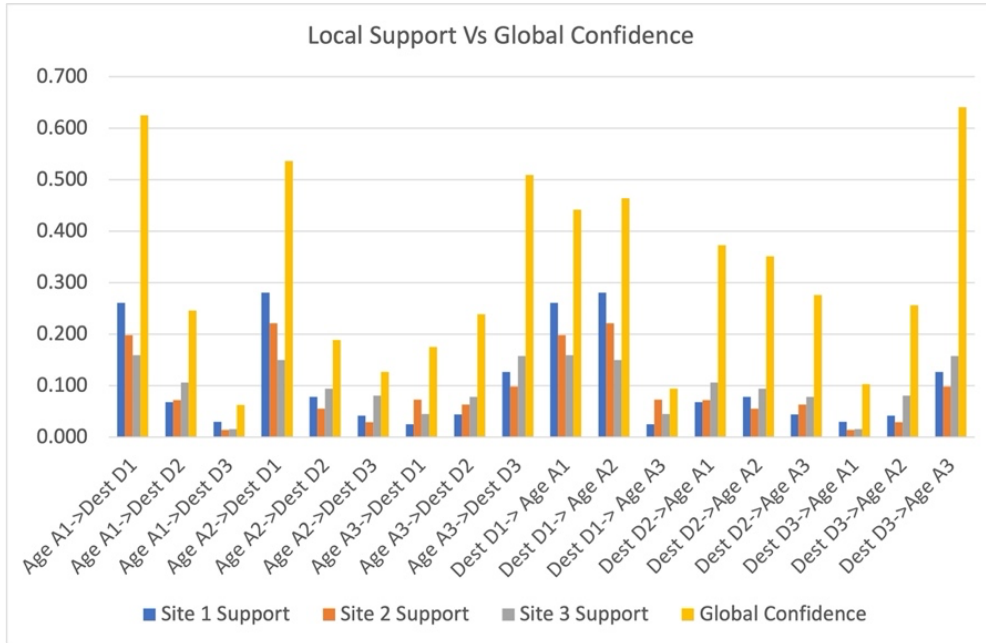


Figure 6.4 Local support count at various sites and global confidence

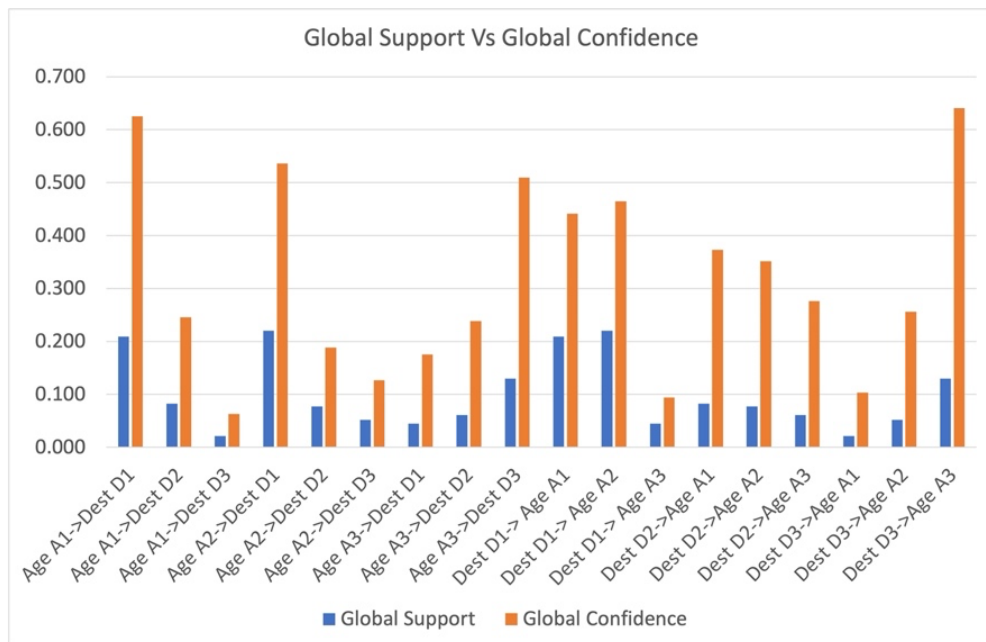


Figure 6.5 Global support count and global confidence

Figure 6.3 shows that global support is proportional to the average of the local supports at different sites whereas global confidence can be high for low as compared to

support count as the case of D3 -> A3 in Figure 6.5. It is observed in Figure 6.4 that support count of A1->D1 is high at site 1 & 2 and not on site 3 but the confidence is high. Whereas support D3-A3 is little high at site 3 only but confidence is quite high. Similarly support D2->A1 is not high at all sites. This shows that the support at each site can't be sufficient to make the global rules. At the same time if the company needs the local association rules for any site, that can be generated from the local frequent itemsets and this may be different from the global rules. Local rules are also important to make local strategies in different states or locations.

This case study developed the distributed data association rule mining model for a tour and travel company. Based on the study many association rules are generated. The results show that the tourists of age group A1 and A2 are more as compared to A3 hence gives the travelling habit based on the age. Similarly, the destination D1 preferred over destination D2 and D3 by the tourists. Whereas destination D3 is visited mostly by age group A3.

6.3 SUMMARY

Distributed association mining based application model for Voyagers Beat, a tour and travel company is proposed and implemented in this chapter. The data from three sites are taken for the analysis. Datasets are transformed for the analysis and association rules between age and destination visited by the tourists are generated. Two experiments are conducted for different minimum support threshold values and further different confidence threshold values. Based on the analysis, interested and useful rules are generated. Relationship between local and global findings are also compared.

7 CONCLUSIONS

This work illustrates some of the algorithms proposed in the literature for mining association rules in centralized and distributed data mining domain and proposes solutions addressing some of the DDM issues. Apriori algorithm is the most popular algorithm for the centralized database. CD and FP-growth[34] are other popular algorithms in data mining. Most of the proposed algorithms in literature used the concept of Apriori, CD, FP-growth[34] algorithms and their variant for finding local frequent itemsets at different sites and used different techniques for further improvement in reducing data scan, storage cost and communication cost. Some of the algorithms and proposed techniques for reducing such costs are illustrated in this work. In different implementations, resources like storage, processing capabilities, communication bandwidth, data volume etc. vary and the challenge is to find the best suited algorithm and trade-off between them for a particular distributed scenario. For mining data, different data mining tools are available with implementation of different data mining algorithms..

In this research, new DARM algorithm QDFIN is proposed for the setup where data partition sizes are not uniform. The proposed algorithm uses the novel data structure, nodeset[85] based on POC tree to generate the local frequent itemsets by each node. It deploys pruning of data to reduce communication load and proposes methodology zero-first technique which balances the load on each node. Execution time performance of the proposed algorithm is evaluated on 4-nodes, 5-nodes, and 6-nodes setup and with varying support threshold. Two experiments are performed, one with uniform and other with non-uniform data partition sizes available on different nodes. The performance of the proposed algorithm is compared with similar latest algorithms FDM and PFIN[19].

The QDFIN outperforms the other two algorithms in the execution time comparisons in all three setups, especially when the data is not uniformly available in all 4, 5, and 6 nodes setups. It scans database only once and creates POC tree. The new zero-first technique is effective in the situation when the data is skewed i.e. data sizes available at various sites differ and not feasible or required to move data from one site to other site to balance the skewness. The performance of the algorithm further improves with the increase of number of sites or nodes as it overcomes the disadvantage of the varying data sizes.

Algorithm SBDARM is proposed for finding frequent itemsets in the distributed data where size of the partitions are varying in size. The proposed algorithm applies the technique of local pruning, no-broadcasting and size based assignment of the polling sites. The algorithm performance is evaluated on four datasets on the 5-nodes setup with varying minimum support threshold. The experiments are performed on the data partitions with varying number of transactions on each site. The execution time of the algorithm is compared with the PFIN and FDM algorithm for distributed data mining.

Algorithm SBDARM outperforms other algorithms on the time of execution comparisons. The new size based assignment technique is effective in the distributed environment where data is generated at different sites and data is skewed. i.e. highly imbalanced in terms of number of transactions at each site. It performs best as the data skewness is not effecting the performance and is well adjusted. In addition to the reduction in candidate sets by pruning, the proposed no-broadcasting technique reduces the communication load and improves the execution time of the proposed algorithm.

Performance of both the proposed algorithms are compared on real and synthetic datasets. The performance of both are very close, but the SBDARM performs better on all

four datasets.

In this research, distributed association rule mining application model is proposed for tour and travel industry to find the interesting facts and relationship between age and destination visited by the tourists. A medium level company is chosen for the study having a few offices in the country. This study can help the company to take decision about opening new booking office and focus on the age of the tourists and adding new destinations. This research may help in expanding the business by adding different destinations, more booking offices in different locations and targeting the potential customers. This study shows that the mining is useful in the tourism industry and can give useful information.

7.1 FUTURE WORK

The research can be further extended for mining in larger setup with more number of sites and large datasets for scalability. The resources and the capabilities available at each site can also be considered while allocating load to different sites for further improvement. It can be further used with the heterogeneous datasets and big data.

There are many more parameters which can affect the tourism industry like weather, gender, customer place of living etc. which can also be considered for mining new facts. The mining can be applied to the bigger sized international company.

PAPERS PUBLISHED

Details of Publications in Scopus Indexed Journals and presented in International Conferences-

- Manoj Sethi and Rajni Jindal, “Distributed association mining for discovering interesting rules for tours and travel company”, 2nd International Conference on Artificial Intelligence: Advances and Applications (ICAIAA 2021), March 2021 (Presented). The after-conference proceeding of the ICAIAA 2021 will be published in Springer Book Series, ‘Algorithms for Intelligent Systems’.
- Manoj Sethi and Rajni Jindal, “Distributed frequent itemset mining using size based assignment technique”, *International Journal of Emerging Trends in Engineering Research*, ISSN: 2347-3983, vol. 8. no. 10, pp. 6765-6773, 2020.
doi: 10.30534/ijeter/2020/248102020 (Scopus).
- Manoj Sethi and Rajni Jindal, “Distributed association rules mining of varying data partition size using Nodesets”, *International Journal of Advanced Trends in Computer Science and Engineering*, ISSN: 2278–3091, vol. 9, no. 4, pp. 5433-54331, 2020.
doi: 10.30534/ijatcse/2020/181942020 (Scopus).
- Manoj Sethi and Rajni Jindal, “Distributed data association rule mining: tools and techniques”, in *Proc. IEEE 10th INDIACom; INDIACom-2016 3rd International Conference on Computing for Sustainable Global Development- BVICAM*, New Delhi, March 2016. (Presented).

REFERENCES

1. A. Ait-Mlouk, F. Gharnati, T. Agouti, "An improved approach for association rule mining using a multi-criteria decision support system: a case study in road safety", *European Transport Research Review*, vol.9, no.40, 2017, doi:10.1007/s12544-017-0257-5.
2. A. Mehay, K. Singh, N. Sharma, "Analyse market basket data using fp-growth and apriori algorithm", *International Journal on Recent and Innovation Trends in Computing and Communication*, vol.1, no. 9, pp. 693-696, 2013.
3. A. Pavithra, and S. Dhanaraj, "Comparative study of effective performance of association rule mining in different databases", in *Proc. of International Conference on Data Science and Analytics, ICDSA 17*, 2017.
4. Ailing Wang, "An improved distributed mining algorithm of association rules", *Journal of Convergence Information Technology*, vol. 6, no. 4, pp.118-122, 2011.
5. Ailing Wang, "Research on mining association rules in distributed system", *International Conference on Business Intelligence and Financial Engineering (IEEE)*, pp. 472-475, 2009, doi: 10.1109/BIFE.2009.113.
6. Alois Ferscha, Janes Johnson and Stephen J. Turner, "Simulation performance data mining", *Elsevier: Future Generation Computer Systems*, vol. 18 pp. 157-174, 2001, doi:10.1016/S0167-739X(01)00050-4.
7. Ammar Alhaj Ali, Pavel Varacha, Said Krayem, Petr Zacek and Andrzej Urbanek, "Distributed data mining systems: techniques, approaches and algorithms", *MATEC Web Conf.*, 22nd International Conference on Circuits, Systems, Communications and Computers (CSCC 2018), 210, 04038, 2018.
8. Andrew Pavlo, Erik Paulson, Alexander Rasin, Daniel J. Abadi, David J. Dewitt, Samuel Madden, Michael Stonebraker, "A comparison of approaches to large-scale data analysis", in *Proc. of the 35th SIGMOD International Conference on Management of Data, (SIGMOD '09)*, pp. 165-178, July 2009, doi:10.1145/1559845.1559865.

9. Asma Belhadi, Youcef Djenouri, Jerry Chun-Wei Lin and Alberto Cano, "A general-purpose distributed pattern mining system", *Springer-Applied Intelligence*, vol. 50, pp. 2647–2662, 2020.
10. Assaf Schuster and Ran Wolff, "Communication-efficient distributed mining of association rules", in *Proc. of the 2001 ACM SIGMOD international conference on Management of data (SIGMOD '01)*, pp. 473-484, May 2001, doi:10.1023/B:DAMI.0000015870.80026.6a.
11. Azra Shamim, Maqbool Uddin Shaikh, Saif Ur Rehman Malik, "Intelligent data mining in autonomous heterogeneous distributed bio databases", in *Proc. of the ICCEA 2010: Second International Conference on Computer Engineering and Applications*, IEEE Computer Society Washington, DC, USA, vol. 1 pp. 6-10, 2010, doi: 10.1109/ICCEA.2010.9.
12. Bagrudeen Bazeer Ahamed and Shanmugasundaram Hariharan, "A survey on distributed data mining process via grid", *International Journal of Database Theory and Application*, vol. 4, no. 3, September, 2011.
13. Bin Liu, Shu-Gui Cao, Qing-Chun Li and Qi Li, "A hierarchical distributed data mining architecture", in *Proc. of the 11th International Conference on Machine Learning and Cybernetics (ICMLC)*, IEEE Guilin, vol. 1, pp. 40-44, July 2011, doi: 10.1109/ICMLC.2011.6016720.
14. Bin Liu, Shu-Gui Cao, Xiao-Li Jia Zhao-Hua Zhi, "Data mining in distributed data environment", in *Proceeding of the Ninth International Conference on Machine Learning and Cybernetics (ICMLC)*, IEEE, pp. 421-426, July 2010, doi: 10.1109/ICMLC.2010.5581024.
15. Bin. Liu, S.G. Cao, W. He, "Distributed data mining for e-business", *Information and Technology and Management*, vol. 12, pp. 67–79, 2011.
16. C. Ju and Dongjun Ni, "Distributed mining model and algorithm of association rules for chain retail enterprise", IEEE: ISECS International Colloquium on Computing, Communication, Control, and Management, 2008, doi: 10.1109/CCCM.2008.129.
17. C. Ju and Dongjun Ni, "Mining frequent closed Itemsets from distributed dataset", International Symposium on Computational Intelligence and Design, IEEE Computer Society, 2008, doi: 10.1109/ISCID.2008.24.

18. C. Wang, Houkuan Huang and Honglian Li, "A fast distributed mining algorithm for association rules with item constraints", IEEE International Conference on Systems, Man, and Cybernetics, Nashville, vol. 3, pp. 1900-1905, TN, 2000, doi: 10.1109/ICSMC.2000.886390.
19. Chen Lin and Junzhong Gu, "PFIN: A parallel frequent itemset mining algorithm using nodesets", *International Journal of Database Theory and Application*, vol. 9, no.6, pp. 81-92, 2016.
20. Chieh-Ming Wu and Yin-Fu Huang, "An effective data structure for mining generalized association rules", IEEE Fifth International Conference on Fuzzy Systems and Knowledge Discovery, 2008, doi: 10.1109/FSKD.2008.609.
21. Dan Hu, Xianchuan Yu and Yuanfu Feng, "Distributed mining core of attributes on horizontally partitioned data", IEEE Pacific-Asia Workshop on Computational Intelligence and Industrial Application, 2008, doi: 10.1109/PACIIA.2008.114.
22. Dan Hu, Xianchuan Yu and Yuanfu Feng, "Distributed mining reducts of attributes on horizontally partitioned data", IEEE Second International Symposium on Intelligent Information Technology Application IITA, 2008, doi: 10.1109/IITA.2008.398.
23. David W. Cheung, Jiawei Han, Vincent T. Ng, Ada W. Fu and Yongjian Fu, "A fast distributed algorithm for mining association rules", in *Proc. of the fourth international conference on Parallel and distributed information systems*, IEEE Computer Society Washington, DC, USA, pp. 31-43, 1996, doi: 10.1109/PDIS.1996.568665.
24. David W. Cheung, Vincent T. Ng, Ada W. Fu and Yongjian Fu, "Efficient mining of association rules in distributed databases", *IEEE Transactions On Knowledge And Data Engineering*, vol. 8, no, 6, pp. 911-922, Dec. 1996, doi: 10.1109/69.553158.
25. Dennis Wegener, Michael Mock, Deyaa Adranale and Stefan Wrobel, "Toolkit-based high-performance data mining of large data on MapReduce clusters", in *Proc. of the 2009 IEEE International Conference on Data Mining Workshops (ICDMW '09)*, pp. 296-301, 2009, doi: 10.1109/ICDMW.2009.34.
26. Dinesh J. Prajapati, Sanjay Garg, N.C. Chauhan, "Interesting association rule mining with consistent and inconsistent rule detection from big sales data in distributed environment"

- Future Computing and Informatics Journal*, vol. 2, no.1, pp.19-30, 2017, doi:10.1016/j.fcij.2017.04.003.
27. E. Cesario and D. Talia, "Distributed Data Mining Models as Services on the Grid", 2008 IEEE International Conference on Data Mining Workshops, Pisa, TBD, Italy, pp. 486-495, Dec. 2008.
 28. E. Cesario, Antonio Grillo, Carlo Mastroianni and Domenico Talia, "A sketch-based architecture for mining frequent items and itemsets for distributed data streams", 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, 2011, doi: 10.1109/CCGrid.2011.45.
 29. F. Bodon and L. Rónyai, "Trie: An alternative data structure for data mining algorithm", *Mathematical and Computer Modelling*, vol. 38, no. 7–9, pp. 739-751, 2003.
 30. Fimi Dataset Repository: Available at <http://fimi.ua.ac.be/data/> .
 31. G. Gatuha and Tao Jiang, "Smart frequent itemsets mining algorithm based on FP-tree and DIFFset data structures", *Turkish Journal of Electrical Engineering & Computer Sciences*, vol. 25, pp. 2096-2107, 2017.
 32. J. Arokia Renjit and K. L. Shunmuganathan, "Mining the Data from Distributed Database Using and Improved Mining Algorithm", *International Journal of Computer Science and Information Security (IJCSIS)*, vol.7, no.3, March 2010.
 33. J. Han, et al., "Frequent pattern mining: current status and future directions", *Data Mining and Knowledge Discovery*, vol. 15, no.1, pp. 55-86, 2007.
 34. J. Han, H. Pei, and Y. Yin, "Mining Frequent Patterns without Candidate Generation" in *Proc. Conf. on the Management of Data (SIGMOD'00, Dallas, TX)*. ACM Press, New York, NY, USA, vol.29, no.2, pp.1-12, 2000.
 35. J. Han, M. Kamber, J. Pie, "Data mining concepts and techniques", in *San Francisco*, Morgan Kaufmann Publishers, 2011.
 36. Jun Liu, Yuan Tian, Yu Zhou, Yang Xiao and Nirwan Ansari, "Privacy preserving distributed data mining based on secure multi-party computation", *Computer Communications*, vol. 153, pp. 208-216, 2020.

37. K. Srikumar, B. Bhasker, "Metamorphosis: mining maximal frequent sets in dense domains", *International Journal on Artificial Intelligence Tools*, vol.14, no.3, pp. 491-506, 2005, doi:10.1142/S0218213005002223.
38. Kouzis-loukas, M., "Analysing Customer Baskets- A Business to Business Case Study", *Financial Economics*, 2014.
39. Kwang-II Ahn, "Effective product assignment based on association rule mining in retail", *Expert Systems with Applications*, vol.39, no.16, pp. 12551–12556, 2012, doi:10.1016/j.eswa.2012.04.086.
40. Liao, Jinggui, Yuelong Zhao and Saiqin Long., "MRPrePost—A parallel algorithm adapted for mining big data", *IEEE Workshop on Electronics, Computer and Applications(IWECA)*, 2014.
41. Lijuan Zhou, Shuang Li and Mingsheng Xu, "Research on Algorithm of Association Rules in Distributed Database System", in *Proc. 2nd International Asia Conference on Informatics in Control, Automation and Robotics (IEEE)*, pp. 216-219, 2010, doi: 10.1109/CAR.2010.5456669.
42. Longyi Li and Lihua Tao, "Study of e-CRM based on Distributed Data Mining", *International Conference on Management and Service Science, MASS 2009*. IEEE Computer Society, 2009, doi: 10.1109/ICMSS.2009.5302106.
43. M. Cannataro, A. Congiusta, Andrea Pugliese and Domenico Talia, "Distributed Data Mining on Grids: Services, Tools and Applications", *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol.34, no.6, 2004, doi: 10.1109/TSMCB.2004.836890.
44. M. E. Otey, C. Wang, S. Parthasarathy, A. Veloso and W. Meira Jr., "Mining Frequent Itemsets in Distributed and Dynamic Databases", in *Proc. of the Third IEEE International Conference on Data Mining (ICDM'03)*, Nov. 2003.
45. M. J. Zaki, S. Parthasarathy, and W. Li, "A localized algorithm for parallel association mining", in *Proc. of the ninth annual ACM symposium on Parallel algorithms and architectures*, ACM, pp. 321–330, 1997.

46. M. Kantarcioglu and C. Clifton, "Privacy preserving distributed mining of association rules on horizontally partitioned data", *IEEE transactions on knowledge and data engineering*, vol.16(9), pp. 1026–1037, 2004.
47. M. Zoubeidi , O. Kazar, S. Benharzallah, N. Mesbahi, A. Merizig, D. Rezki, "A new approach agent-based for distributing association rules by business to improve decision process in ERP systems", *International Journal of Information and Decision Sciences*, vol. 12, no.1, pp. 1-35, 2020.
48. Manpreet Kaur and Shivani Kang, "Market Basket Analysis: Identify the changing trends of market data using association rule mining", International Conference on Computational Modelling and Security (CMS 2016) in *Procedia Computer Science* 85, pp. 78 – 85, 2016 doi:10.1016/j.procs.2016.05.180.
49. Merve Er Kara, Seniye Ümit Oktay Fırat, Abhijeet Ghadge, "A data mining-based framework for supply chain risk management", *Computers & Industrial Engineering*, vol. 139, 105570, 2020, doi:10.1016/j.cie.2018.12.017.
50. Mohammed J. Zaki, "Parallel and distributed association mining: a survey", *Journal of IEEE Concurrency*, vol. 7, no. 4, pp. 14-25, October 1999, doi: 10.1109/4434.806975.
51. Mohammed. J. Zaki, "Parallel and distributed data mining: An introduction", in *Large-Scale Parallel Data Mining, Lecture Notes in Computer Science*, vol.1759, Springer, Heidelberg, pp.1–23, 2000. doi: 10.1007/3-540-46502-2_1.
52. N. Isa, N.A. Kamaruzzaman, M.A. Ramlan, N.Mohamed, M.Puteh, "Market Basket Analysis of Customer Buying Patterns at Corm Café", *International Journal of Engineering and Technology*, vol. 7, no. 4.42, 2018, doi:10.14419/ijet.v7i4.42.25692.
53. Nader Aryabarzan, Behrouz Minaei-Bidgoli and Mohammad Teshnehlab, "negFIN: An efficient algorithm for fast mining frequent itemsets", *Expert System and Applications*, vol. 105, pp. 129-143, 2018.
54. Norulhidayah Isa, Nur Syuhada Mohd Yusof, Muhammad Atif Ramlan, "The Implementation of Data Mining Techniques for Sales Analysis using Daily Sales Data", *International Journal of Advanced Trends in Computer Science and Engineering*, vol. 8, no. 1.5, pp. 74 – 80, 2019, doi:10.30534/ijatcse/2019/1681.52019.

55. NVS Pavan Kumar, Dr. J K R Sastry and Dr. K Raja Sekhara Rao, “Mining distributed databases for negative associations from regular and frequent patterns”, *International Journal of Advanced Trends in Computer Science and Engineering*, vol. 8, no. 4, pp. 1449-1463, 2019.
56. NVS Pavan Kumar, JKR Sastry and K Raja Sekhara Rao, “Mining negative frequent regular itemsets from data streams” *International Journal of Emerging Trends in Engineering Research*, vol. 7(8), pp. 85 – 98, 2019.
57. P. Naresh and Dr. R. Suguna, “Association rule mining algorithms on large and small datasets: a comparative study”, in *Proc. International Conference on Intelligent Computing and Control Systems (ICICCS 2019)*, IEEE: CFP19K34-ART, pp. 587-592, 2019.
58. Pallavi Dubey, “Association Rule Mining on Distributed Data” *International Journal of Scientific & Engineering Research*, vol. 3, no. 1, Jan. 2012.
59. Pyun, Gwangbum, Unil Yun, and Keun Ho Ryu, "Efficient frequent pattern mining based on linear prefix tree", *Knowledge-Based Systems*, vol. 55, pp. 125-139, 2014.
60. R. Agrawal and R. Srikant, "Fast algorithms for mining association rules", in *Proc. International Conference on Very Large Scale Data Base*, 1994, pp. 487-499.
61. R. Agrawal, and C. Shafer John, "Parallel mining of association rules", in *IEEE Transactions on Knowledge & Data Engineering*, vol. 8, Issue 6, pp. 962-969, 1996.
62. R. Agrawal, T. Imielinski , A. Swami, “Mining association rules between sets of items in large databases”, In *Proc. of International Conference of ACM-SIGMOD on Management of Data*, pp. 207-216, 1993.
63. R. Moodley, F. Chicilana, F. Caraffini, J. Carter, “Application of Uninorms to Market Basket Analysis”, *International Journal of Intelligent Systems*, vol.34, no.1, pp. 39-49, 2019, doi:10.1002/int.22039.
64. S. Bhagwat, V. Jethliya, A. Pandey, L. Islam, “Sales analysis using product rating in data mining techniques”, *International Journal of Research in Engineering and Technology*, vol. 4, no.2, 189-191, 2015.

65. S. Rathee, M. Kaul and A. Kashyap, "R-Apriori: an efficient apriori based algorithm on spark", in *Proc. of the 8th workshop on Ph.D. workshop in information and knowledge management, PIKM 15*, Melbourne: ACM, pp. 27–34, 2015.
66. Sanket Thakare, Sheetal Rathi, and R. R. Sedamkar, "An improved Prepost algorithm for frequent pattern mining with Hadoop on cloud", in *Procedia Computer Science*, vol. 79, pp. 207-214, 2016.
67. Sotiris Kotsiantis and Dimitris Kanellopoulos, "Association Rules Mining: A Recent Overview", *GESTS International Transactions on Computer Science and Engineering*, vol.32, no.1, pp. 71-82, 2006.
68. Sudarsan Biswas, Neepta Biswas, Kartick Chandra Mondal, "Parallel and distributed association rule mining algorithms: a recent survey", *Information Management and Computer Science (IMCS)*, vol. 2(1), pp. 15-24, 2019
69. Sujni Paul, "An Optimized Distributed Association Rule Mining Algorithm in Parallel and Distributed Data Mining with XML Data for Improved Response Time", *International Journal of Computer Science and Information Technology*, vol.2, no.2, 2010.
70. V. Devasekhar, and P. Natarajan, "Multi-agent distributed data mining: challenges and research directions" *International Journal on Emerging Technologies*, vol. 11, no.4, pp. 233–239, 2020.
71. V. Gancheva, "Market basket analysis of beauty products", M.S. Thesis, Erasmus School of Economics, Erasmus University Rotterdam, 2013.
72. Van Quoc Phuong Huynh and Josef Küng, "FPO Tree and DP3 algorithm for distributed parallel frequent Itemsets mining", *Expert Systems With Applications*, vol. 140, 112874, 2020.
73. Van Quoc Phuong Huynh, Josef Küng and Tran Khanh Dang, "A Parallel incremental frequent itemsets mining IFIN⁺: improvement and extensive evaluation", *Transactions on Large-Scale Data-and Knowledge-Centered Systems XLI*, pp. 78-106, 2019.
74. Van Quoc Phuong Huynh, Josef Küng, Markus Jäger and Tran Khanh Dang, "IFIN⁺ a parallel incremental frequent itemsets mining in shared- memory environment", in *Proc. International Conference on Future Data and Security Engineering FDSE 2017*, pp. 121-138, 2017, doi: 10.1007/978-3-319-70004-5_9.

75. Vasoya A., Koli N., "Mining of association rules on large database using distributed and parallel computing" *Procedia Computer Science*, 79, pp. 221 – 230, 2016.
76. Vinaya Sawant and Ketan Shah, "Performance evaluation of distributed association rule mining algorithms", 7th International Conference on Communication, Computing and Virtualization, *Procedia Computer Science* 79, pp. 127-134, 2016.
77. Vuda Sreenivasarao, Rallabandi Srinivasu, Prof. G.Ramaswamy, Nagamalleswara Rao Dasari and Dr. S Vidyavathi, "The Research of Distributed Data Mining Knowledge Discovery Based on Extension Sets", *International Journal of Computer Applications (0975 – 8887)*, vol.8, no.2, Oct. 2010, doi: 10.5120/1187-1658.
78. Wei Fan and Albert Bifet, "Mining Big Data: Current Status and Forecast to the Future", *ACM SIGKDD Explorations Newsletter*, vol. 14, no. 2, pp. 1-5, Dec. 2012, doi:10.1145/2481244.2481246.
79. Wenxiang Dou, Jinglu Hu, Kotaro Hirasawa and Gengfeng Wu, "Distributed Multi-Relational Data Mining Based on Genetic Algorithm", *IEEE Congress on Evolutionary Computation*, pp. 744-75, 2008, doi: 10.1109/CEC.2008.4630879.
80. Wu-Shan Jiang and Ji-Hui Yu, "Distributed Data Mining on the Grid", in *Proc. of the fourth International Conference on Machine Learning and Cybernetics*, IEEE, 2005, doi: 10.1109/ICMLC.2005.1527275.
81. Xindong Wu, X. Zhu, G. Wu and W. Ding, "Data mining with big data", *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 1, Jan. 2014, doi: 10.1109/TKDE.2013.109.
82. Y. Kurnia, Y. Isharianto, Yo Ceng Giap, A. Hermawan, Riki, "Study of application of data mining market basket analysis for knowing sales pattern (association of items) at the O! Fish restaurant using apriori algorithm", *Journal of Physics: Conference Series*, vol.1175 012047, 2018.
83. Z. Ashrafi Mafruz, D. Ashrafi, D. Taniar, and K. Smith, "ODAM: An Optimized Distributed Association Rule Mining Algorithm" *Distributed Systems Online*, IEEE, vol.5, no. 3, 2004, doi: 10.1109/MDSO.2004.1285877.
84. Zhi-Hong Deng, "DiffNodesets: An efficient structure for fast mining frequent itemsets", *Applied Soft Computing*, vol. 41, pp. 214-223, 2016.

85. Zhi-Hong Deng, and Sheng-Long Lv., "Fast mining frequent itemsets using Nodesets", *Expert Systems with Applications*, vol. 41, no. 10, pp. 4505-4512, 2014.
86. Zhi-Hong Deng, and Sheng-Long Lv., "PrePost+: An efficient N-lists-based algorithm for mining frequent itemsets via children-parent equivalence pruning", *Expert Systems with Applications*, vol. 42, no.13, pp. 5424-5432, 2015.
87. ZhiHong Deng, ZhongHui Wang, and JiaJian Jiang, "A new algorithm for fast mining frequent itemsets using N-lists", *Science China Information Sciences*, vol.55, no. 9, pp. 2008-2030, 2012.