A FRAMEWORK FOR ASSOCIATION RULE MINING OF DISTRIBUTED DATA

Ph.D. THESIS

By

GURPREET SINGH BHAMRA
(Regn. No. 90703508)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
THAPAR UNIVERSTIY PATIALA 147004 (INDIA)

JUNE, 2015

A FRAMEWORK FOR ASSOCIATION RULE MINING OF DISTRIBUTED DATA

A THESIS

Submitted in partial fulfillment of the
Requirements for the award of the degree
of

DOCTOR OF PHILOSOPHY

in

COMPUTER SCIENCE AND ENGINEERING

By

GURPREET SINGH BHAMRA
(Regn. No. 90703508)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
THAPAR UNIVERSTIY PATIALA 147004 (INDIA)

JUNE, 2015

Candidate's Declaration

I hereby certify that the work being presented in the thesis entitled **"A Framework for Association Rule Mining of Distributed Data"** in fulfillment of the requirements for the award of the degree of **"Doctor of Philosophy"** and submitted in Computer Science and Engineering Department, Thapar University, Patiala, is an authentic record of my own work carried out under the supervision of **Dr. Anil Kumar Verma**, Associate Professor, Thapar University, Patiala and **Dr. Ram Bahadur Patel**, Professor, Chandigarh College of Engineering & Technology, Chandigarh and refers other researchers works which are duly listed in the reference section.

The matter presented in this thesis has not been submitted for the award of any other degree of this or any other university.

(Gurpreet Singh Bhamra)
Regn. No. 90703508

This is to certify that the above statement made by the candidate is correct and true to the best of our knowledge and belief.

Date:

(Dr. Anil K. Verma)
Associate Professor,
TU, Patiala

(Dr. Ram B. Patel)
Professor,
CCET, Chandigarh

Abstract

The exponential rise in the collected data generated an essential need for new techniques that can convert this huge amount of data into useful knowledge. Consequently, Data Mining (DM) has become a powerful technology focusing on the most important information in the massive data. DM extracts the interesting data patterns from large databases using computational techniques/tools. The classical central data warehouse(DW) based DM approach is ineffective or infeasible because of heavy storage, computational and communication costs involved in managing data from the ever increasing and privacy-sensitive distributed resources. Distributed Data Mining (DDM) is emerged as an active sub-area of DM research. DDM is concerned with application of classical DM procedures in a distributed computing environment to effectively utilize the available resources.

Association Rules (ARs) are used to discover the associations among frequent itemsets in a database. Association Rule Mining (ARM) today is one of the most important aspects of DM task. In ARM all the strong association rules are generated from the frequent itemsets. Distributed Association Rule Mining (DARM) generates the globally strong association rules from the global frequent itemsets in a distributed environment for the global decision making.

Agent mining also known as Agent enriched DM, is an emerging interdisciplinary area that integrates agent technology, DM, machine learning. Most of the existing agent based frameworks for DARM task are only prototype model and lacks the appropriate underlying Agent Execution Environment (AEE), scalability, privacy preserving techniques, global knowledge and implementation using a real datasets especially in bio-informatics domain. Bio-informatics or computational molecular biology aims at automated analysis and the management of high-throughput biological data as well as modeling and simulation of complex biological systems.

Mining the ARs from the frequent itemsets requires a transactional dataset which can be a real transactional datset of any retail industry or can be a synthetic version generated by a tool. A software tool called Transactional Dataset Generator (TDSG) has been designed and implemented in Java language for generating a synthetic dataset.

Traditional central DW based approach for ARM is practically investigated with the help of a client-server based framework. The overall response time for the ARM task performed using this approach is also formulated. The outcome of this approach suggested

the use of agent technology for DARM task for the issue of scalability and global knowledge extraction.

An AEE is designed and implemented that acts as a distributed server application for managing a multi-agent system (MAS) for DARM task. It provides the appropriate functionality to Mobile Agents (MAs) to execute, communicate, migrate to other platform, manage itinerary and use system resources.

A scalable MAS called Agent enriched Mining of Globally Strong Association Rules (AeMGAR) that act as framework for DARM task is designed and implemented using two computing models. In a serial computing model MAs visit n distributed sites serially and performs their designated tasks. Global knowledge and performance of this system is compared with the traditional central DW based ARM approach. Serial itinerary used for MAs increases the overall cost of DARM task so a parallel computing model is designed. Clones of MAs in parallel computing model visit n distributed sites in parallel and it is found that overall response time for the DARM task involving n distributed sites is very less in case of parallel computing model of AeMGSAR. The comparative analysis on various parameters reveals that the proposed AeMGSAR framework has improved features and exhibit superior performance than the existing agent based DARM frameworks.

As mining biological data is an emerging area at the intersection between DDM and bio-informatics, we have also taken the case of DARM in bio-informatics and designed another version of this framework called Agent enriched Quantitative Association Rules Mining for Amino Acids in distributed Protein Data Banks (AeQARM-AAPDB) for mining the quantitative ARs for amino acids in proteins. Experimental tests on real data have confirmed its effectiveness. A comparative analysis on various parameters shows that the proposed system outperforms existing model.

This thesis may be considered as an approach that advocates the integration of MAS and DM especially in bio-informatics. A scalable agent based framework for ARM of distributed data has been designed and implemented and further enhanced as a case study in bio-informatics.

Keywords: Association Rules, Distributed Association Rule Mining, Mobile Agent, Multi-Agent System, Bio-informatics

To

GOD

ਆਪਹਿ ਕੀਆ ਕਰਾਇਆ ਆਪਹਿ ਕਰਨੈ ਜੋਗੁ ॥
ਨਾਨਕ ਏਕੋ ਰਵਿ ਰਹਿਆ ਦੂਸਰ ਹੋਆ ਨ ਹੋਗੁ ॥੧॥
– ਗੁਰੂ ਗ੍ਰੰਥ ਸਾਹਿਬ – ਅੰਗ ੨੫੦

He Himself acts, and causes others to act;
He Himself can do everything.
O Nanak, the One Lord is pervading everywhere;
there has never been any other, and there never shall be. ||1||

– Shri Guru Granth Sahib, Page 250

Acknowledgements

A few sublime human experiences defy expressions of any kind, and a feeling of true gratitude is one of them. I, therefore, find words quite inadequate to express my gratitude to my supervisors **Dr. A. K. Verma**, Associate Professor, Thapar University, Patiala and **Dr. R. B. Patel**, Professor, Chandigarh College of Engineering & Technology, Chandigarh for their virtuous guidance, support and encouragement throughout this work. Their deep insight into the problem and the ability to provide solutions has been immense value in improving the quality of research at all stages. This experience of working with them shall ever remain a source of inspiration and encouragement for me. I learned a great deal from them, not only about research but also matters touching many other aspects that will benefit me in my future endeavors.

My sincere thanks are due to Sh. Tarsem Garg, Chancellor, Maharishi Markandeshwar University, Mullana and Dr. Prakash Gopalan, Director, Thapar University, for providing me necessary administrative assistance in completion of the work.

I express my heartfelt thank to Dr. Deepak Garg, Head, Computer Science & Engineering Department, for his guidance and supportive attitude. I express my gratitude to the Doctoral Committee members Dr. Maninder Singh, Dr. Rajesh Kumar and PG Coordinator for monitoring the progress and providing valuable suggestions for improvement of my Ph.D. work from time to time. I thank all faculty members of Department of Computer Science and Engineering, Thapar University for their kind hearted support.

I am extremely grateful to the celebrated authors whose precious works have been consulted as a guiding light and referred in my research work. I also wish to convey my appreciation to fellow research scholars and colleagues who provided encouragement and timely support in the hour of need.

At the end I express my deep sense of appreciation to my wife Manjot and children Shabadpreet and Gurleen for their cooperation and patience during the completion of my research work. My parents have given me their unequivocal support throughout, as always, for which my mere expression of thanks likewise does not suffice.

All the thanks are, however, only fraction of what is due to Almighty for granting me an opportunity and the divine grace to successfully accomplish this assignment.

GURPREET SINGH BHAMRA

Contents

List of Abbreviations

AA	Amino Acids
AAFFA	Amino Acids Frequency Finder Agent
AEE	Agent Execution Environment
AeMGSAR	Agent enriched Mining of Globally Strong Association Rules
AeQARM-AAPDB	Agent enriched Quantitative Association Rules Mining for Amino Acids in distributed Protein Data Banks
AFARMDD	Agent based Framework for Association Rules Mining of Distributed Data
AL	Agent Launcher
AR	Association Rule
ARM	Association Rules Mining
BDS	Binary Dataset
BODHI	Besiezing knOwledge through Distributed Heterogeneous Induction
CDWM	Central Data Warehouse Manager
CORBA	Common Object Request Broker Architecture
DAI	Distributed Artificial Intelligence
DAME	Distributed Agent based Mining Environment
DARM	Distributed Association Rules Mining
DC	Distributed Computing
DCOM	Distributed Component Object Model
DD	Dataset Dispatcher
DDM	Distributed Data Mining
DECAF	Distributed, Environment Centered Agent Framework
DKN	Distributed Knowledge Networks
DM	Data Mining
DM_AEE	Data Mining Agent Execution Environment

DSM	Dataset Merger
DW	Data Warehouse
EJB	Enterprise Java Beans
EMADS	Extendible Multi-Agent Data mining System
FI	Frequent Itemset
FIM	Frequent Itemset Mining
FIPA	Foundation for Intelligent Physical Agents
FISCG	Frequent Itemsets and Support Counts Generator
FMIDBGA	Frequency Mapping and Itemset Data Bank Generater Agent
GFI	Global Frequent Itemset
GFIGA	Global Frequent Itemset Generater Agent
GKB	Global Knowledge Base
GKDA	Global Knowledge Dispatcher Agent
GKDA_P	Global Knowledge Dispatcher Agent with Parallel itinerary
GKGA	Global Knowledge Generater Agent
JADE	Java Agent DEvelopment framework
JAM	Java Agents for Meta-Learning
JDBC	Java Database Connectivity
JVM	Java Virtual Machine
LFI	Local Frequent Itemset
LFIGA	Local Frequent Itemset Generater Agent
LKB	Local Knowledge Base
LKCA	Local Knowledge Collecter Agent
LKCA_P	Local Knowledge Collecter Agent with Parallel itinerary
LKGA	Local Knowledge Generater Agent
LKGA_P	Local Knowledge Generater Agent with Parallel itinerary
MA	Mobile Agent
MADARM	Multi-Agent Distributed Association Rule Miner
MADKDS	Mobile Agent based Distributed Knowledge Discovery System
MAS	Multi-Agent System
OIKI	Optimized Incremental Knowledge Integration
PADMA	PArallel Data Mining Agents
PDB	Protein Data Bank
PDBFA	Protein Data Bank Filtering Agent
RIGKGA	Result Integration and Global Knowledge Generater Agent
RM	Result Manager
RMI	Remote Method Invocation

SARH	Strong Association Rule Harvester
TDS	Transactional Dataset
TDSG	Transactional Dataset Generator
TFICA	Total Frequent Itemset Collector Agent
TID	Transactional ID
WMS	Workflow Management System

List of Tables

List of Figures

List of Algorithms

Chapter 1

INTRODUCTION

During the last few decades, the exponential rise in the collected data generated an essential need for new technique that can wisely transform this enormous amount of data into useful knowledge. Consequently, Data Mining (DM) has become a prominent technology to focus on the most important information hidden in the massive data. DM is the process of extraction of interesting patterns or trends from huge databases [42, 53]. It involves the use of computational techniques/tools which include algorithms, statistical models and machine learning techniques. The classical approach for knowledge discovery in distributed environment takes a single centrally integrated data repository called Data Warehouse (DW). DM techniques are applied on DW to mine the data and extract the knowledge [91].

The organisation of this chapter is as follows. The issues involved with the central DW based DM approach are discussed in Section 1.1. and Section 1.2 presents an overview of Distributed Data Mining (DDM) and the issues involved. Association Rule Mining (ARM), Apriori algorithm and Distributed Association Rule Mining (DARM) is highlighted in Section 1.3. Preliminaries notations used throughout the thesis are also highlighted. In Section 1.4 a general description of agents, multi-agent system (MAS) and role of agents in DDM/DARM is provided. Section 1.5 provides an overview of bioinformatics and protein sequences. Motivations for this research task are given in Section 1.6. Research objectives are briefly written in Section 1.7. Section 1.8 highlights the contribution of the thesis. The organization of the thesis is discussed in Section 1.9 and finally the chapter is summarized in Section 1.10.

1.1 Issues with Central DW based approach

The central DW based approach is not suitable because of heavy storage and computational costs involved in managing data from the ever increasing and updated distributed resources where data is produced continuously in streams. Network communication cost is also involved in transferring huge data over the wired or wireless network where network bandwidth is a limitation. It is also not desirable to centrally collect the privacy-sensitive raw distributed data of the business organizations viz, banking, and telecommunication. Modern business organizations data are geographically distributed and horizontally or vertically fragmented. It is difficult to combine such data at a central location. Performance and scalability of DM applications may be increased by distributing the workload

among sites [86, 109]. Distributed and mobile environment has certain resource constraints which are not considered in algorithms for central DW based DM as certain data sets are immovable in practice [71]. Therefore, it is evident that the traditional DM approach is not suitable for dealing with todays distributed environments and applications mining the privacy-sensitive data.

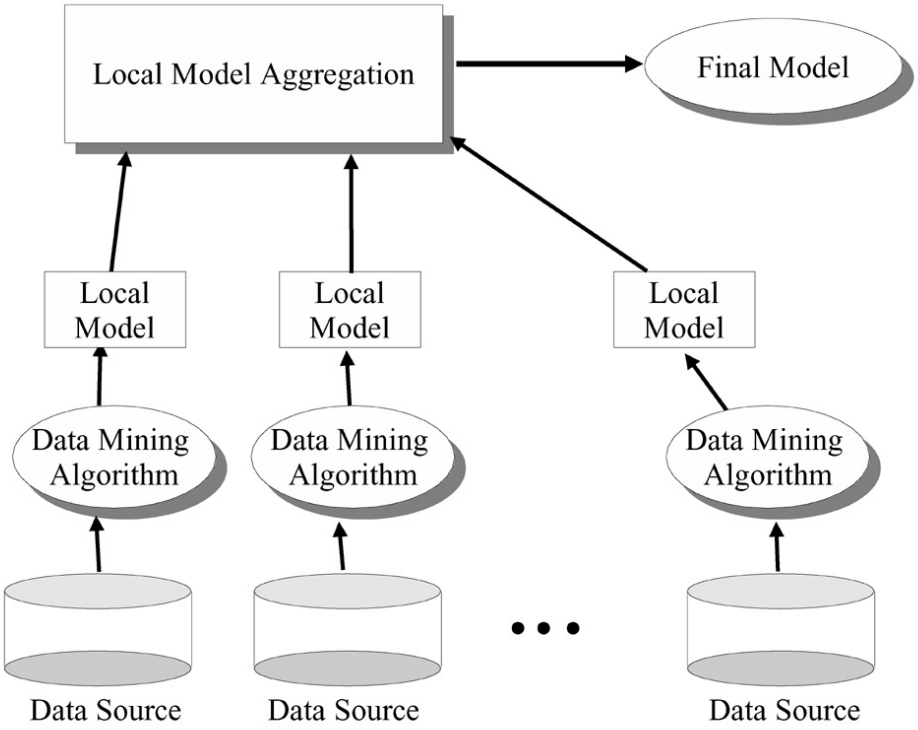

Figure 1.1: Distributed Data Mining Framework [91]

1.2 Distributed Data Mining

The issues discussed in the previous section for centralized DW based DM approach results into the development of techniques for DDM. It is concerned with application of classical DM procedures in a DC environment trying to make the best of the available resources including communication networks, computing units and distributed data repositories, human factors, etc. In DDM, DM takes place both locally as well as globally. At a local level DM takes place at each geographically distributed site and at a global level DM takes place at a central site where the local knowledge is merged in order to discover global knowledge. The main aim in DDM is to get the global company-wide knowledge for decision making from the local operational data at distributed sites. On-line, real-time decision support based distributed applications can be designed in DDM. Authors in [91] provide a broad and updated overview of DDM techniques.

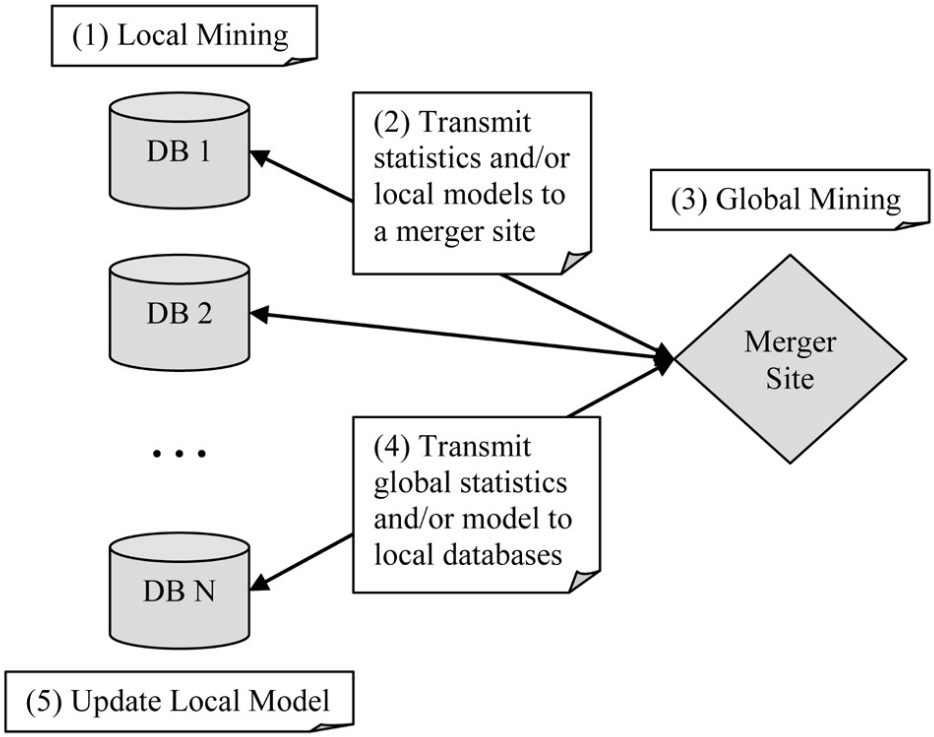

Figure 1.2: Typical architecture of Distributed Data Mining approaches [109]

In one of the important general DDM architecture {Fig. 1.1} proposed in [91], processing at the different distributed nodes generates several local models which are then aggregated to form a global model representing the global knowledge. The type and availability of distributed resources are taken into account while performing the DM operations. Authors in [109] proposed another phase wise DDM approach {Fig. 1.2}. In the first phase local distributed databases are analyzed. The locally discovered knowledge is transmitted to a central merger site, where all the local models are integrated. The global knowledge is transmitted back to update the distributed databases.

1.2.1 Issues with DDM

A DDM system is comprised of many components viz. mining algorithms, distributed resources, communication channels, task scheduling, and user interfaces, etc. The distributed data and computing resources in a DDM system must be accessed efficiently. Monitoring of entire mining task and presentation of results in suitable formats also required. A DDM system should also be adaptable, i.e., flexible enough to adjust the various situations. Based upon the type and size of the given resources, it should dynamically identify the best possible mining approach.

The data which are stored on distributed sites, over networks of heterogeneous and autonomous source, are growing rapidly. This ever growing data poses several problems for knowledge discovery. The major challenges for any DDM approach are concerned with the scalability, autonomy, privacy protection of the local data, data replication, data fragmentation and communication cost minimisation [44]. Much research effort has been

done to address these issues. Most of the approaches in these efforts are based on the agent technology [44, 65, 86].

1.3 Association Rule Mining

Let $DB = \{T_j, j = 1 \dots D\}$ be a transactional dataset of size D with transaction identifier (TID) for every transaction (T) and $I = \{d_i, i = 1 \dots m\}$, total m data items in DB. A set of items in a particular transaction T is called itemset or pattern. An itemset $P = \{d_i, i = 1 \dots k\}$, which is a set of k data items in a particular transaction T and $P \subseteq I$, is called k-itemset. Support of an itemset, $s(P) = \frac{No\_of\_T\_containing\_P}{D}\%$, is the frequency of occurrence of itemset P in DB, where $No\_of\_T\_containing\_P$ is the support count (sup_count) of itemset P. Frequent itemsets (FIs) are the itemsets that appear in DB frequently, i.e., if $s(P) \geq min\_th\_sup$ (given minimum threshold support), then P is a Frequent k-Itemset. They play an essential role in mining the interesting relationships among itemsets. Frequent Itemsets Mining (FIM) is the task to find the set of all subsets of FIs in a transactional database. It is the processor intensive task, mainly because of the large amount of itemsets generated and large size of the datasets involved [53].

Association Rules (ARs) first introduced in [3], are used to discover the associations (or co-occurrences) among items in a database. These are used to find the customers purchase patterns where it is decided as to how the buying some goods will impact on the transactions of buying others. Such rules become helpful to design the goods shelves and to manage the stock. Customers can also be classified according to the purchase patterns using these rules. AR is an implication expression of the form $P \Rightarrow Q[support, confidence]$ where , $P \subset I$,$Q \subset I$ and P and Q are disjoint itemsets, i.e., $P \cap Q = \emptyset$. The support and confidence are two measuring unit for an AR where:

- Support $s(P \Rightarrow Q) = p(P \cup Q) = \frac{No\_of\_T\_containing\_both\_P\_and\_Q}{D}\%$: the probability of both P and Q appearing in T, we can say that s % of the transactions support the rule$P \Rightarrow Q$, $0 \leq s \leq 1.0$ or $0\% \leq s \leq 100\%$
- Confidence $c(P \Rightarrow Q) = p(Q \mid P) = \frac{s(P \Rightarrow Q)}{s(P)} = \frac{sup\_count(P \Rightarrow Q)}{sup\_count(P)}\%$: the conditional probability of Q given P, we can say that when itemset P occurs in a transaction there are $c\%$ chances that itemset Q will occur in that transaction, $0 \leq c \leq 1.0$ or $0\% \leq c \leq 100\%$

An Association rule $P \Rightarrow Q$ is said to be strong if

1. $s(P \Rightarrow Q) \geq min\_th\_sup$, i.e., support of the rule is greater than or equal to the given minimum threshold support and
2. $c(P \Rightarrow Q) \geq min\_th\_conf$, i.e., confidence of the rule is greater than or equal to the given minimum threshold confidence

If the attributes or items in a database have boolean values then this is called *boolean association rules problem* where presence of the item is marked by 1 and absence by 0. Attributes in database can be quantitative (e.g. income, age, frequency of amino acids in protein) or categorical (e.g. make of car, zip code). In this *quantitative association rules problem* domain, the attribute values are finely partitioned into ranges (intervals) and

then each $< attribute, interval >$ pair is mapped to a Boolean value. Authors in [104] discusses the problem of mining ARs in large relational tables containing both quantitative and categorical attributes. ARM today is one of the most important phase of DM process. In ARM all the strong association rules are generated from the FIs. The ARM process involves two steps [4, 119].

1. Find list of all frequent k-itemsets
   $L = \{L_k, k = 1 \dots p | \ no\ further\ frequent\ itemsets\ are\ generated\ after\ p\}$.
2. Generate strong ARs $\forall (L_k | k > 1) \in L$
   (a) Generate all non empty subsets of frequent itemset l, $\forall l \in L_k$
   (b) For every non empty subset s of l, output the rule "$s \Rightarrow (l - s)$", if $\frac{sup\_count(l)}{sup\_count(s)} \geq min\_th\_conf$

The first step, i.e., FIM step determines overall performance of ARM task and the second step is much less costly.

1.3.1 Apriori Algorithm for mining Frequent Itemsets

FIM, being the first step in ARM task, has lots of proposed algorithms both sequential and parallel. Since ARM is dedicated to handle the huge amount of data so time complexity and resource complexity in such algorithms are carefully considered. Apriori algorithm is proposed by R. Agrawal and R. Srikant in 1994 for mining FIs for boolean ARs [5]. Once a set of FIs are obtained, ARs can be straightforwardly generated from FIs, so Apriori has emerged as one of the best FIM and subsequently ARM algorithms. Many subsequent ARM-related algorithms are designed based on Apriori algorithm. An iterative candidate-generation-and-subset-test approach is followed in Apriori algorithm for idenfifying FIs in data. When all the subsets in a candidate itemset are identified as frequent in the previous pass then it is confirmed as frequent itemset. It is directly based upon an important property called the Apriori property. It says that *All nonempty subsets of a frequent itemset must also be frequent.*

First of all, the list of frequent 1-itemsets (L_1) is obtained by scanning the database and applying the constraint of given minimum threshold support. Next, an iterative search approach is followed where list of frequent k-itemsets (L_k) is used to discover list of frequent (k+1)-itemsets (L_{k+1}) ,i.e., L_1 is used to discover L_2, which is further used to discover L_3, and so on, until no more frequent k-itemsets exist. It requires one full scan of the database to find each L_k. This algorithm scans the database k times (size of the largest frequent itemset) even when only one frequent k-itemset exists. This factor contributes one of the limitations of this algorithm for a very large size database.

Many algorithms using different types of approaches have been proposed in the literature to improve the efficiency of Apriori algorithm. Comparative analysis of the few of them can be found in [25, 52, 56, 60, 103, 119]. Many existing algorithms are efficient only for small size dataset. When database is huge, the core data structure (e.g. the list of candidates in Apriori) of such algorithms also becomes unmanageable and shows runtime error. Real datasets are applications based and can be distributed, massive, dynamic, light and dense. Distributed applications need more scalability. There is no single

FIM/ARM algorithm that can satisfy all such requirements [8]. Recent research in designing FIM/ARM algorithm are focusing on dynamic datasets, minimizing communication cost and fault tolerance.

1.3.2 ARM Example

Analysis of the shopping basket is a common example of ARM where customer's buying habits are analysed by checking the correlations among the various items. This analysis helps retailers to design policies by looking into which items are frequently purchased together. If customers purchase potato chips and coke together, then to increase the sale of both items coke is placed near to potato chips.

Table 1.1: Market Basket Transaction Data

TID	Items
1	milk coke cereal
2	sausage mustard coke cereal
3	cereal butter bread
4	milk cigarettes cereal butter bread
5	sausage nappies milk cereal bread
6	milk cereal bread
7	nappies cigarettes cereal candy butter
8	nappies mustard cereal candy
9	sausage mustard coke butter bread
10	coke candy bread

Table 1.1 shows a small dataset of shopping basket transactions. Each transaction has their Transaction ID (TID). The minimum threshold support has been set to 0.2 or 20%. The steps for generating ARs for the data shown in Table 1.1 by applying the Apriori algorithm are described below. Each identified set of frequent itemsets is shown within curly brackets along with the support.

- 1^{st} **scan:** for finding frequent 1-itemsets
  - {bread}:0.6,{butter}:0.4,{candy}:0.3, {cereal}:0.8, {cigarettes}:0.2, {coke}:0.4, {milk}:0.4, {mustard}:0.3, {nappies}:0.3, {sausage}:0.3
- 2^{nd} **scan:** for discovering frequent 2-itemsets
  - **Candidate 2-itemsets:** all pairs of the frequent 1-itemsets.
  - **Frequent 2-itemsets:** {butter, bread}:0.3, {cereal, bread}:0.4, {coke, bread}:0.2, {milk, bread}:0.3, {sausage, bread}:0.2, {cereal, butter}:0.3, {cigarettes, butter}:0.2, {cereal, candy}:0.2, {nappies, candy}:0.2, {cigarettes, cereal}:0.2, {coke, cereal}:0.2, {milk, cereal}:0.4, {mustard, cereal}:0.2, {nappies, cereal}:0.3, {sausage, cereal}:0.2, {mustard, coke} :0.2, {sausage, coke}:0.2, {sausage, mustard}:0.2
- 3^{rd} **scan:** for discovering frequent 3-itemsets

  - **Candidates 3-itemsets:** {cereal, butter, bread}:0.2, {coke, cereal, bread}:0.0, {milk, cereal, bread}:0.3, {sausage, cereal, bread}:0.1, {sausage, coke, bread}:0.1, {cigarettes, cereal, butter}:0.2, {nappies, cereal, candy}:0.2, {mustard, coke, cereal}:0.1 {sausage, coke, cereal}:0.1, {sausage, mustard, cereal}:0.1, {sausage, mustard, coke}:0.2
  - **Frequent 3-itemsets:** {cereal, butter, bread}:0.2, {milk, cereal, bread}:0.3, {cigarettes, cereal, butter}:0.2, {nappies, cereal, candy}:0.2, {sausage, mustard, coke}:0.2

- 4^{th} **scan:** for discovering frequent 4-itemsets
  - Further candidates are not found

After finding all the FIs, ARs can be straightforwardly derived from those. Examples of few of the ARs are given below:

- **FIs** {cigarettes, cereal, butter}:0.2 and {cereal, butter}:0.3
- **AR** {cereal, butter} ⇒ {cigarettes}
  - Support of AR: 0.2 or 20%
  - Confidence of AR: 0.2/0.3=66%
- **FIs** {milk, cereal, bread}:0.3 and {milk, bread}:0.3
- **AR** {milk, bread} ⇒ {cereal}
  - Support of AR: 0.3 or 30%
  - Confidence of AR: 0.3/0.3 = 100%
- **FIs** {cereal, butter}:0.3 and {butter}:0.4
- **AR** {butter} ⇒ {cereal}
  - Support of AR: 0.3 or 30%
  - Confidence of AR: 0.3/0.4 = 75%

1.3.3 Distributed Association Rule Mining

Distributed Association Rule Mining (DARM) generates globally strong ARs from the global FIs in a distributed environment. Data remains unevenly balanced among partitions in distributed database. It becomes desirable to mine the global rules for the global decisions. The local rules are used for the local decisions.

1.3.3.1 Preliminaries and Definitions

Few preliminary notations and definitions required for defining DARM and to make this study self-contained are as follows:

- $S = \{S_i, i = 1 \dots n\}$, n distributed sites.
- $S_{CENTRAL}$, Central Site.
- $DB_i = \{T_j, j = 1 \cdots D_i\}$, Horizontally partitioned Data Set of size D_i at the local site S_i, where each transaction T_j is assigned an identifier(TID).
- $DB = \bigcup_{i=1}^{n} DB_i$, the Aggregated Dataset of size $D = \sum_{i=1}^{n} D_i$, $DB_i \cap DB_j = \varnothing$
- $I = \{d_i, i = 1 \dots m\}$, total m data items in each DB_i.
- $L_{k(i)}^{FI}$, Local Frequent k-itemsets at site S_i.
- $P_{sup\_count}^{i}$, Support count of an itemset P in DB_i at site S_i. Itemset P is locally frequent at site S_i, if $P_{sup\_count}^{i} \geq min\_th\_sup \times D_i$.
- $L_{k(i)}^{FISC}$, List of support count $\forall Itemset \in L_{k(i)}^{FI}$.
- L_i^{LSAR}, List of locally strong association rules at site S_i.
- $L^{TLSAR} = \bigcup_{i=1}^{n} L_i^{LSAR}$, List of total locally strong association rules.
- $L_k^{TFI} = \bigcup_{i=1}^{n} L_{k(i)}^{FI}$, List of total frequent k-itemsets.
- $L_k^{GFI} = \bigcap_{i=1}^{n} L_{k(i)}^{FI}$, List of global frequent k-itemsets such that $L_k^{GFI} \subseteq L_k^{TFI}$
- P_{sup_count}, Support count of an itemset P in DB. Itemset P is globally frequent, if $P_{sup\_count} \geq min\_th\_sup \times D$.
- $L_{CENTRAL}^{GSAR}$, List of Globally strong association rule.

Local Knowledge Base (LKB), at site S_i, comprises of $L_{k(i)}^{FI}$, $L_{k(i)}^{FISC}$ and L_i^{LSAR} which can provide reference to the local supervisor for local decisions. Global Knowledge Base (GKB), at $S_{CENTRAL}$, comprises of L^{TLSAR}, L_k^{TFI}, L_k^{GFI} and $L_{CENTRAL}^{GSAR}$ for the global decision making [116]. Like ARM, DARM task can also be seen as a two-step process [119]:

1. Find the Global frequent k-itemset (L_k^{GFI}) from the distributed Local Frequent k-Itemsets($L_{k(i)}^{FI}$) from partitioned datasets.
2. Generate globally strong association rules ($L_{CENTRAL}^{GSAR}$) from L_k^{GFI}.

1.4 Software Agent Technology

Intelligent software agent technology is an interdisciplinary technology inherited from different research disciplines and sub-fields such as, Distributed Computing (DC), Distributed Artificial Intelligence (DAI), advanced knowledge base systems, distributed information retrieval systems, and human computer interaction. The motivating idea of agent technology is the development and efficient utilization of autonomous software entities, called intelligent software agents or intelligent agents, which have access to geographically distributed and heterogeneous information resources [23, 69]. Agent research stems from the work in DAI conducted in the 1970s. An Actor system is proposed in [55] where each Actor had an explicit internal state and had the capability to respond to the messages of other Actors. The subsequent years witnessed the focus on the more theoretical aspect of bringing intelligence to software agents, whereas the last decade has seen a huge expansion of systems to solve practical problems due to advances in programming, distributed resources, the internet, and the increased digitization of information and services. Intelligent agents simplify the complexities of DC [48]. There would be increasing demand of large number of interacting components in the form of services or intelligent agents in the next generation of computing systems [76].

1.4.1 Taxonomies and Characteristics

Researchers have proposed number of classifications of agents and criteria required for being an intelligent software agent. Agents with different attributes are being designed depending upon the goals to be achieved and responsibilities to be carried out. Authors in [85] identified three primary attributes, autonomy, cooperation, and learning and classified the intelligent agents as collaborative, collaborative learning, interface, and smart. Based upon the mobility, temporal continuity, reactivity, role played, hybrid philosophies, kindness, trustworthiness, and mentalistic and emotional qualities, agents can be classified as mobile agents, information/internet agents, reactive agents, and hybrid agents [40, 43, 115]. Numerous definitions for the agents have been proposed, but in core most have a set of defining characteristics that every agent must possess. These capabilities are listed below.

- *Autonomous*: Autonomy is the capacity of operating without the intervention of users, i.e., capability of modifying the way in which they achieve their objectives (situation awareness). The degree of autonomy and authority is called agency of an agent. The nature of communication between the agent and various other components in the system is a qualitative measuring unit of agency. At a minimum level of agency, an agent must run a-synchronously and at the enhanced level it represents a user/human in some way.

- *Reactive or Deliberative*: the capacity of perceiving any change in the environment and suitably reacting to it. The current state of an agent decides which action should be undertaken.

- *Adaptive*: the capacity of sensing the environment dynamically and reconfiguring thereafter. This can be achieved by embedding alternative problem solving rules/strategies or algorithms for the agent's choice.

- *Pro-activeness*: the capacity of showing goal-oriented behavior in order to satisfy a design objective.
- *Intelligence*: the degree of reasoning and learned behavior.
- *Temporal continuity*: the capacity of perseverance of identity and state over long time period.
- *Social, Cooperative, Collaborative*: the ability of interacting and negotiating with other agents in groups and/or human users to achieve a common goal via some kind of agent communication language. They cooperate, negotiate and collaborate with other agents to make a smart system.
- *Flexibility*: the ability of exhibiting reactivity, pro-activeness and social ability simultaneously.
- *Mobility/Migration/Navigation*: the ability of moving from a node to node in a heterogeneous network. Mobility can be of weak and strong in nature. In weak mobility, currently idle (not running) agent moves to another machine, i.e., only agent code is transferred and the agent starts its execution from the starting point on the remote machine. In strong mobility, an agent who is currently performing its task (running) moves to another machine, i.e., agent code as well execution state (data stack) is transferred and the agent continues its calculations/execution from the point of suspension on the remote machine like it was never interrupted. Mobility raises security and efficiency issues.

1.4.2 Mobile Agent

An intelligent software agent with mobility feature is known as Mobile Agent (MA). MA is an autonomously transportable code that migrate itself from one network node to another without losing its operability. In other words, a program's running state is suspended at a host and it is transferred to another host and resume its suspended state there. An agent execution environment on the next node verify its authenticity and it resume its execution from point of suspension. To accomplish its task, MA can create child or clone agents or interact with other stationary agents. After completion of the journey It submits the results to the sending client. It resembles a human agent, working for clients by visiting a place, using a service/resource and then moving on to next destination. Various attributes of a MA are i) Identification to identify a MA and its dispatching station, ii) Itinerary consisting of the number and order of the hosts to be visited by MA, iii) Data unit containing the required data, iv) Code unit containing the transportable code, v) Execution state, and vi) External state for intermediate results [94, 95, 96, 113].

A Multi-Agent System (MAS) is distributed application/system composed of multiple interacting and coordinating intelligent agent components [114]. An Agent Execution Environment (AEE) or Agent Development Tool, is a middleware distributed server application that provides the appropriate functionality to software agents to authenticate, execute, communicate, migrate to other platform, and use system resources in a secure way. An AEE is mainly used to provide a framework with higher level of abstraction for the development, execution and management of MAS without going into the low level

communication details and just focusing on the desired functioning of the agents themselves [9, 27, 40, 50].

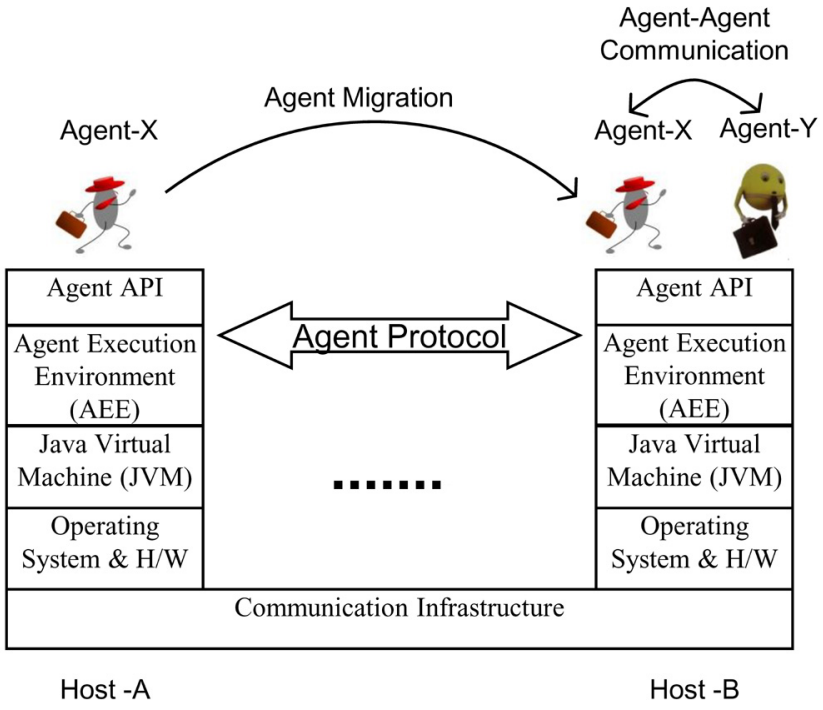

Figure 1.3: General architecture of a java based AEE

The conceptual model of a Java based AEE is depicted in Fig. 1.3. It consists of a) Host machine or a Network Node with Operating System and other hardware; b) Java Virtual Machine (JVM)- a runtime environment for platform independence and security; c) AEE - A multi-threaded server application running on the top of JVM and provides basic functionality for agent management, migration, communication and security etc., in the form of java agent packages; and d) Communication Infrastructure (wired or wireless network) connecting the Host machines. Java MAs are special Java objects, executed as threads within an AEE. TCP/IP is used as the main transport protocol.

1.4.2.1 General benefits of MAs

MAs are an exciting area of research within the ever growing intelligent software agent arena. They can be considered as a replacement of the traditional client-server model of distributed computing. They have become commercially viable with recent technologies and have the potential for revolutionizing network applications. They are naturally heterogeneous and goal oriented in nature and roam around in an unstructured network to gather information. MAs provide the following key advantages:

- *Reduction in network bandwidth and latency*: The amount of data that can be transferred over the network in a given time is known as bandwidth of the network and latency (response time delay) indicates the time for a signal to travel from one point to another in the network. MAs efficiently and economically use the communication channels having low bandwidth and high latency. If a large amount

of data is involved in transfer, the consumption of network bandwidth can be high, with a resultant delay in response time or latency. With MA, a single serialized object is transmitted over the network carrying the resultant data only, reducing the consumption of network bandwidth and latency.

- *Reduction in network traffic*: They reduce the network traffic because only resultant data is carried over the network to a central site and unnecessary intermediate results transmission is avoided.

- *Robust and fault-tolerant*: Since the MA needs only to be transported once between two sites, the possibility of failure due to network faults is reduced and for this reason, mobile agents are especially appropriate for wireless network links. MA may have the intelligence to bypass the faulty hosts or seek temporary shelter on a reliable host.

- *Disconnected operation*: MA can continue to function in an unstructured network even if it is disconnected from its dispatching host (home) and send back results upon reconnection. This feature is particularly useful for low-cost, light weight, portable computing devices having the low processing powers, memory constraints, and intermittent low bandwidth connection to the main network of such mobile device.

- *Load balancing*: Some of the processor and memory intensive tasks of a program may be shifted to other more powerful and lightly loaded platforms due to strong mobility feature of an agent. This plays a significant role in case of low power portable mobile device and speed up programs.

- *Scalable, configurable and reusable*: MAs can be viewed as the building blocks of the distributed applications. A distributed application can be designed as a collection of MA components. Such an application can easily be extended with new functionality in the form of agents with slight modifications to the master agent. They are self-contained and reusable. Number of participating hosts can be increased without any significant impact on the complexity of the application.

- *Cloning*: The parent agent can also clone several child agents. These clones implement concurrent operations, and improves running efficiency.

- *Rapid prototyping*: New versions of the software components of distributed applications can be dynamically deployed in the form of or using MAs.

The qualitative strengths of intelligent autonomous agents make current agent platforms an attractive choice for a wider and wider range of distributed applications. Agents may be deployed in different settings and application domains such as Information retrieval system; Information fusion [45]; Network management; Mobile computing; Cloud computing [101]; Workflow and Administrative Management; Electronic commerce; Industrial control; Internet searching; Global software development [99] and distribution; Personal assistance; Language information system [98]; Games; Protein function prediction [108] in Bio-informatics; Clinical data analysis for medical diagnosis; Distributed data mining; Image similarity [61]; Intrusion detection system; Financial forecasting [13];

Human Computer interactions [39]; Peer-to-Peer computing; Grid and cluster computing; Wireless sensor network; Unmanned air and under-water vehicles; Oil production systems etc.

Table 1.2: Single Letter codes for Amino Acids

Sr. No.	Amino Acid Name	Single Letter Code	Three Letter Code
1	Alanine	A	Ala
2	Cysteine	C	Cys
3	Aspartic Acid	D	Asp
4	Glutamic Acid	E	Glu
5	Phenylalanine	F	Phe
6	Glycine	G	Gly
7	Histidine	H	His
8	Isoleucine	I	Ile
9	Lysine	K	Lys
10	Leucine	L	Leu
11	Methionine	M	Met
12	Asparagine	N	Asn
13	Proline	P	Pro
14	Glutamine	Q	Gln
15	Arginine	R	Arg
16	Serine	S	Ser
17	Threonine	T	Thr
18	Valine	V	Val
19	Tryptophan	W	Trp
20	Tyrosine	Y	Tyr

1.5 Bio-informatics

Bio-informatics or computational molecular biology aims at automated analysis and the management of high-throughput biological data as well as modeling and simulation of complex biological systems. In a very broad sense it solves the problems in molecular biology with use of mathematical and computer science models. Bio-informatics has very much changed since the first sequence alignment algorithms in 1970s [47]. Today in-silico analysis is a fundamental component of biomedical research. Bio-informatics has now encompasses a wide range of subject areas from structural biology, genomic to gene expression studies [14, 31, 74, 90, 100].

Cellular machinery of any organism comprises of proteins and functioning of proteins heavily depends upon the amino acid sequence present in it. The functioning of protein might completely change with a slight change in the amino acid sequence. A proteins sequence comprises of 20 types of Amino Acids (AA). Each AA is represented by a single alphabet code as shown in Table 1.2. Such AA sequence completely decides unique 3-dimensional structure of each protein.

Most of the existing agent based frameworks for DARM task are only prototype model and lacks the appropriate underlying AEE, scalability, privacy preserving techniques, global knowledge and implementation using a real dataset, especially in bio-informatics domain. The scalability is particularly an important issue as the amount of data is increasing rapidly in a distributed scenario. A DM system is said to be scalable if it operate effectively without any degradation in its performance on increasing number of data sites. Parallel or DDM is the potential solution to the scalability issue. As mining biological data is an emerging area at the intersection between bio-informatics and DDM, we have also taken the case of DARM in bio-informatics for mining the quantitative ARs for amino acids in distributed protein data banks.

1.6 Motivation

The important features of MA being autonomous, scalable, capable of reducing network bandwidth and latency, capable of handling intermittent network connections motivated us to design and implement a scalable agent based framework for DARM task.

The nature of associations between different amino acids has been a subject of great importance. Amalgamation of bio-informatics, DDM and MAS also motivated to design a agent based framework for mining quantitative ARs among amino acids in proteins.

1.7 Objectives

The aim of the thesis is to design and evaluate a scalable agent based framework or system for DARM task. Stating in brief the objectives of the thesis are as follows:

1. To explore the existing techniques and frameworks for Association Rule Mining of Distributed Data.
2. To design and develop a framework for Association Rule Mining of Distributed Data using Mobile Agents.
3. To develop an environment for the proposed framework.
4. Validation and verification of the proposed framework.

1.8 Work Carried Out

As per objectives, we have designed and implemented a scalable framework called Agent enriched Mining of Globally Strong Association Rules (AeMGSAR) for DARM task. It integrates a case study for mining real biological data. To achieve the **first objective**, a rigorous review of literature has been done and found that the existing agent based frameworks for DARM task are only prototype models. Scalability, design of an effective AEE, privacy preserving techniques for the local data, global knowledge extraction and implementation using a real dataset especially in bio-informatics domain are open issues. Here we have reviewed agent based frameworks for DARM task in general and MAS in bio-informatics. The scalability is particularly an important issue as the amount of data

is increasing rapidly in a distributed scenario especially in bio-informatics domain. Major contributions of this thesis in line with objectives of thesis, are discussed briefly as follows:

- **Tool for synthetic dataset generation:** Mining the Association rules from the frequent itemsets requires a transactional database which can be a real transactional data base of any retail industry or can be a synthetic version generated by a tool. Synthetic dataset generated by a tool can serve a fundamental requirement for experimenting with the DM concepts and mining the ARs from the frequent itemsets. A software tool called Transactional Dataset Generator (TDSG) for generating a Binary Dataset (BDS) as well as Transactional Dataset (TDS) corresponding to the Binary Dataset has been designed for this purpose.

- **Central DW based ARM:** Traditional central DW based approach for ARM is practically investigated. The outcome of this approach suggested the use of agent technology for DARM task for the issue of scalability and global knowledge extraction.

- **Agent Execution Environment for DARM:** To fulfill the requirement of **third objective**, a generic AEE is designed and implemented that acts as a distributed server application for managing a MAS for DARM task. It provides the appropriate functionality to MAs to execute, communicate (with other agents, users, and other platforms), migrate to other platform, manage itinerary and use system resources (local and global knowledge).

- **Agent based framework for DARM:** To achieve **second objective**, a scalable framework called Agent enriched Mining of Globally Strong Association Rules (AeMGSAR) for DARM task is designed and implemented. Serial as well as parallel computing model of this system are designed based on the itinerary pattern of MAs. To accomplish **fourth objective** we have implemented the framework as well as runtime environment in Java and the performance of the proposed framework is compared with the traditional central DW based ARM approach and other existing DARM frameworks. The comparative analysis on various parameters reveals that the proposed AeMGSAR framework has improved features and exhibit superior performance than the existing agent based DARM frameworks.

- **A Case study in Bio-informatics:** As mining biological data is an emerging area at the intersection between bio-informatics and DDM, we have also taken the case of DARM in bio-informatics and designed another version of this framework called Agent enriched Quantitative Association Rules Mining for Amino Acids in distributed Protein Data Banks (AeQARM-AAPDB) for mining the quantitative ARs for amino acids in proteins. The entire system is implemented in Java and a comparative analysis on various parameters shows that the proposed system outperforms existing model.

1.9 Organization of the Thesis

Including introductory chapter this thesis is organized into six chapters.

Chapter 2 presents a critical review of state of the art in agent technology in DDM / DARM especially in bio-informatics. The chapter primarily identifies the gaps in existing frameworks for DARM task in general and for bio-informatics in specific. Identified gaps are taken as opportunities for work in next chapters.

Chapter 3 practically investigates the traditional central DW based approach for ARM. The outcome of this approach suggested the use of agent technology for DARM task for the issue of scalability and global knowledge extraction.

Chapter 4 presents a scalable system called Agent enriched Mining of Globally Strong Association Rules (AeMGSAR) for DARM task. Performance of the system is compared with the traditional central DW based ARM approach. This system integrates an AEE which acts as a distributed server application for managing a MAS for DARM task.

In Chapter 5 a case of DARM task in bio-informatics is taken and another version of this system called Agent enriched Quantitative Association Rules Mining for Amino Acids in distributed Protein Data Banks (AeQARM-AAPDB) is presented. This system helps for mining the quantitative ARs for amino acids in proteins.

Finally, Chapter 6 summarizes the presented work. It also briefly highlights about the contributions and future scope.

1.10 Summary

The necessary background for the research task is described in this chapter. It includes the distributed data mining, software agent technology and bio-informatics, research motivations, finalised objectives, and the work carried out for achieving these objectives. The next chapter presents a literature review relevant to the research task.

Chapter 2

LITERATURE REVIEW

This chapter presents literature review of the current research relating to agent based DARM. The chapter is divided into two main sections, Section 2.1 reviews the agent technology in DDM/DARM. Existing DDM systems and comparative analysis of agent based DARM system along with issues involved are also discussed. Section 2.2 reviews the use of software agent technology in bio-informatics domain and existing MAS in this domain are also discussed. A discussion on the issues of existing agent based system in bio-informatics is highlighted in Section 2.2.2 and finally summary of the chapter is presented in Section 2.3.

2.1 Review of Agent Technology in DDM/DARM

The problems and challenges of DDM as discussed in section 1.2 of Chapter 1 and inherent features of software agents clearly indicate the use of MA technology for developing advanced DDM systems. Agent mining, also known as Agent enriched Data Mining is an emerging interdisciplinary area. It integrates MAS, DM and knowledge discovery, machine learning and statistics [26]. One or more agents are deployed per data site in all of the agent based DDM system. These agents mine the local data and communicate with other agents. A global knowledge is generated by merging and analysing the locally mined knowledge. The use of agent technology in DDM is motivated by two factors. Firstly, DDM technology has welldefined sub-modules, the need to encapsulate different mining algorithms into a single unit, the requirement for interaction and co-ordination between different components and the ability to deal with the distribution of resources. These characteristics are naturally suitable for an agent based approach. Secondly, agent technology effectively address scalability and enhance the performance of the system by reducing computational, communication and storage costs associated with the transfer of large volumes of data on the network. In the second criterion for using agents as the building blocks of DDM systems, the focus is on the mobility aspects of agency in addition to the collaborating and information aspects [70, 88].

2.1.1 Existing DDM systems

Three architectural frameworks are reported in the literature for the development of DDM systems. These are the client-server model, the agent-based model and the hybrid ap-

proach which integrates the previous two techniques. Common Object Request Broker Architecture (CORBA), Distributed Component Object Model (DCOM), Enterprise Java Beans (EJB), Remote Method Invocation (RMI) and Java Database Connectivity (JDBC) are the important technologies used to develop client-server DDM [88]. Kensington Enterprise Data Mining Decision Centre [29], IntelliMiner [93] and InterAct[92] are few of the client-server based DDM systems.

In recent years, a number of DDM solutions are provided using various techniques such as, distributed clustering, classification, compression, and distributed association rules but only a few of them make use of intelligent agents [68]. The agent based model can make use of MAs and stationary agents [70]. These systems are generally Java based to support the need for heterogeneity and platform independence. PArallel Data Mining Agents (PADMA)[66], Java Agents for Meta- Learning (JAM) [106], Besiezing knOwledge through Distributed Heterogeneous Induction (BODHI) [64], Papyrus[16], InfoSleuth[80], Distributed Knowledge Networks (DKN) [58], a mediator oriented agent based DDM system [15], Optimized Incremental Knowledge Integration (OIKI) [10], Extendible Multi-Agent Data mining System (EMADS)[6, 7, 8] are the most important DDM systems developed using agent-based approach.

Authors in [70] compared DecisionCentre, IntelliMiner, InterAct, PADMA, JAM, BODHI, Papyrus and InfoSleuth DDM systems and proposed a hybrid model, integrating the client-server and mobile agent model, called Distributed Agent based Mining Environment (DAME) for delivering internet-based DDM services. Authors in [107] proposed a Foundation for Intelligent Physical Agents (FIPA)-compliant platform called Agent Academy for creating and managing a MAS.

2.1.2 Comparative analysis of agent based DARM systems

The existing agent based systems specifically dealing with the DARM task are: Mobile agent based distributed knowledge discovery system (MADKDS)[110], Efficient Distributed Data Mining using Intelligent Agents [1], Mobile Agent based Distributed Data Mining [71], An Agent based Framework for Association Rules Mining of Distributed Data (AFARMDD)[59], Multi-Agent Distributed Association Rule Miner (MADARM)[87]. All these systems are academic research projects. Discussion of these and few others are given below.

Authors in [110] proposed a mobile-agent based distributed knowledge discovery system (MADKDS). DM agent encapsulates a novel incremental algorithm, IAA[111] for mining the local frequent itemsets at distributed sites and collect this knowledge at central DW which results into an increase in the storage cost at central site. Underlying AEE is designed using IBM Aglet Workbench [73]. Parallel itinerary is maintained for MAs and this framework is implemented using Java and C++ as dynamic link library through Java native interface. No privacy preserving techniques are used for the local knowledge. No user interface for the MAS is designed. No cost model for the overall DARM task is discussed and experimental validation using a large size synthetic or real data set is also required.

An Agent based Framework for Association Rules Mining of Distributed Data (AFARMDD) is proposed in [59]. This study mainly addresses the issue of privacy protection of the local data. It encapsulates the existing privacy preserving techniques proposed in

[30, 62] into agents. Encryption and decryption of secure union and secure sum operations at each site are encapsulated into various DM agents. Parallel as well as serial itinerary is maintained for MAs. Agent Server and Local Host components are designed as underlying AEE. Apriori [40] algorithm is used for mining the local frequent itemsets. Privacy preserving techniques are discussed for the local sensitive data and these techniques are the core area of this study. No user interface for the MAS is designed. No cost model for the overall DARM task is discussed and experimental validation using a large size synthetic or real data set is also required. Globally strong association rules are also not generated.

Authors in [87] proposed theoretical cost models for agent based DARM task using a prototype model called Multi-Agent Distributed Association Rule Miner (MADARM). These cost models are basically used to predict the response time of a DARM task. Specific agents are designed for coordination, performing ARM task, migrating and integrating the results. Theoretical cost models are the core area of this study. Apriori [5] and FP-growth [54] algorithms are considered for mining the local frequent itemsets. Parallel as well as serial itinerary is discussed for MAs. No underlying AEE is discussed. Only conceptual views are presented in the study and the authors conclude that the work still needs improvement and experimental validation.

In an experimental setup, authors in [1] performed efficient DDM intelligent agents incorporating standard Apriori [5] algorithm implemented in J#. Though authors claim that it is an agent based setup but it has been observed that there is no AEE and related agents exist in the study. So lot of work needs to be done designing an agent based framework where intelligent agents are actually implemented comprising a MAS on the top of an AEE using a large synthetic or real datasets.

Authors in [71] proposed another agent based DDM approach to reduce the time required to compute Global Frequent Itemset (GFI). No information is given about which algorithm used by Mining Agent to generate Local Frequent Itemset (LFI). No AEE exist in the study. Implementation, validation and the underlying AEE required to actually perform the agent enabled DDM using a large synthetic or real datasets.

Table 2.1: Qualitative comparison of three Agent based DARM Frameworks

Features	**DARM Framework**		
	MADKDS[110]	**AFARMDD**[59]	**MADARM**[87]
Itinerary	Parallel	Serial and Parallel	Serial and Parallel
AEE	Yes	Yes	No
Impl	Yes	Yes	No
Algorithm	IAA [111]	Apriori [5]	Apriori [5], FP-growth [54]
CM	No	No	Yes
PP	No	Yes	No
GUI	No	No	No
DS	No	No	No
Use	No	No	No

Qualitative comparison of some prominent current agent based DARM frameworks is provided in Table 2.1 taking into account some of the features they provide. The features

include the following fields:

- *Itinerary* indicates the serial or parallel travel plan followed by mobile agents.
- *AEE* shows whether an underlying Agent Executing Environment is used for developing MAS.
- *Impl* field indicates whether the MAS is implemented along with the language used for implementation or it is just a prototype framework.
- *Algorithm* indicates the FIM/ARM algorithm considered in the study.
- *CM* indicates whether any cost model is discussed in the study.
- *PP* points out whether any privacy- preserving mechanism is taken into account for the sensitive local data.
- *GUI* is for the Graphical User Interface feature of the MAS.
- *DS* indicates whether any dataset (synthetic or real) is used in experimental validation.
- *Use* indicates use of framework in practical applications, development projects, case studies etc.

This analysis reveals that AFARMDD[59] and MADARM[87] is based on the parallel and serial itinerary of the MAs whereas MADKDS[110] use parallel itinerary. Only MADKDS[110] has an existing IBMs Aglet Workbench as the underlying AEE, others dont have it to test and validate the DARM system. MADARM[87] is only a prototype model without any implementation. Apriori [5] algorithm is mostly used for FIM in such systems. Only MADARM[87] discuss the cost model involved in the entire DARM task, others dont have it. Privacy preserving mechanism for the sensitive local data is the core area of AFARMDD [59] system while others dont have any such mechanism. None of these frameworks has any Graphical user interface designed to work with the systems. None of these frameworks is used in any real applications like mining biological database in bio-informatics.

2.1.3 Issues with agent based DARM

Most of the existing agent based frameworks for DARM task are only prototype model and lacks the appropriate underlying AEE, scalability, privacy preserving techniques, global knowledge extraction, GUI, cost models and implementation using a real dataset, especially in bio-informatics domain. The scalability is particularly an important issue as the amount of data is increasing rapidly in a distributed scenario. A DM system is said to be scalable if it operate effectively without any degradation in its performance on increasing number of data sites. Cost model for the overall DARM task should be effectively addressed. Such systems must be equipped with a case study usage. Researchers in this area should focus more on developing architecture and algorithms that will reduce the massive data movement in global knowledge mining thereby reducing the response time. Further algorithms and methods should also consider the development of adaptive, fault tolerant

and easily extendable systems in the area of DARM. So lot of work still needs to be done designing an agent based framework where intelligent agents are actually implemented comprising a MAS on the top of an AEE using large synthetic or real datasets.

The next section reviews the use of software agent technology in bio-informatics domain.

2.2 Review of Agent Technology in Bio-informatics

The amalgamation of the DDM and MAS provides rewarding solution in terms of security, scalability, storage cost, computation cost and communication cost. Mining biological data is an emerging area at the intersection between bio-informatics and DM. It continues to be an extremely important problem, both for DM research and for biomedical sciences [118].

A Workflow is the coordinated execution of multiple tasks or activities in a process applying a set of procedural rules. A software system called Workflow Management Systems (WMS) is used to defines, creates and manages the execution of workflows [57, 112]. The biological experiment can also be seen as a workflow as bio-scientists performs their in-vitro experiments or in-silico daily work by executing a set of distinct, time consuming and repetitive activities in a distributed environment [82, 105]. Academic research oriented and industrial WMSs have already been proposed and are being applied in the biomedical domain [97].

The concept of DM came to biology around 2000. Since then lots of DM algorithms, software tools, and web tools on bio-informatics have been proposed and designed. Software agents have been considered for DM from 2001. AgentLink, funded by the European Commission's 6^{th} Framework Programme [2], was a Coordination Action for Agent Based Computing for providing support for researchers and developers in the domain of agent-based computing. Agent technology provides an appropriate solution for applications that deals with the repetitive, time-consuming, interactive and coordinated activities; learning, planning and knowledge management and modelling and simulations of complex, dynamic systems [34, 102]. The advantages of software agents, discussed in section 1.4.2.1, make the agent technologies and MASs suitable for bio-informatics field and constitute an active and emerging area both in relation to designing domain-aware MASs for simulating and modeling biological systems through autonomous components interactions; and for the automation of data collection and service discovery processes along with knowledge management and problem-solving. Interactions among agents in a MAS resembles to the interactions among entities in the modeled biological system. Biological components and their environment show both reactive and proactive behavior that can be directly modelled as agent abstraction [77, 81]. The genome scientists can design reagents for future research in livestock genomic with the help of communities of software agents. Software agents are being used to assist in the process of submitting sequence data to GenBank at the Meat Animal Research Center at Clay Center, Nebraska [67]. Agent technology would be a promising technology for biological data analysis in near future [117].

2.2.1 MAPs and MASs in Bio-informatics

Use of intelligent software agent technology in bio-informatics started around 2000. Most of the research in bio-informatics using agent technology focuses on designing MAS for WMS [18]. Some prominent MAS in bio-informatics domain are GeneWeaver[24], BioMAS[37, 36], BioAgent[82], CellMAS[35, 83], BIOPACMAS[89], Agent based framework for protein structure prediction (PSP) [22], MAS for gene expression data [72], Multi-agent based bio-data mining [117], MAS for Next Generation Laboratory Information Management Systems [78]. Description of some of the systems are given below.

GeneWeaver [24] is a prototype MAS designed for the genome analysis and PSP. It composed of a community of task oriented agents. Each agent encapsulates the existing databases and tools and interact with other agents to effectively automate the biological processes. Each agent shares a common AEE comprising the generic modules of control, motivation, action, interaction, and communications. It is a prototype system only. Implementation and validation of the system still need to be done using an efficient execution environment.

BioMAS [36, 37] is a prototype MAS based on Distributed, Environment Centered Agent Framework (DECAF) [49] toolkit for the genome annotation and sequencing for herpesvirus. It uses the distributed and open nature of its multi-agent solution to expand the information gathering system in several ways that will make it useful for biologists studying more organisms, and in different ways. Implementation and ontology support for MAS is still required.

BioAgent [82]is a MAS to support bio-scientists during the process of genome analysis and annotation. The 4-layered software architecture of *BioAgent* system consists of the Core layer, Service agents layer, Bio agents layer, and Workflow layer. It is implemented using Java language. System is constantly evolving by adding more functionality and has to prove itself in a complex real application

PAC-MAS[89] is a Personalized, Adaptive, and Cooperative MAS designed to predict secondary structure of protein. It is a generic multi agent architecture consisting of four main levels: information, filter, task, and interface. Each level is associated to a specific role and populated by a society of agents. It is a prototype system only. Implementation and validation of the system still need to be done using an efficient execution environment.

CellMAS [35, 83] is a MAS designed on the top of *Hermes* [21, 32, 33, 34] for modeling carbohydrate oxidation in cellular process. *Hermes* is a 3-layered, component-based, AEE completely developed in Java for design and execution of workflow based distributed applications. It is a simulation model for which verification and validation is still required.

Authors in [22] presented agent based framework for PSP problem implemented using the Prolog and C++. Each amino acid is represented by independent software agent that communicates with the others. An efficient AEE is required for this simulation model.

In [72] authors proposed a multi-agent approach to conduct the analysis of gene expression data generated using micro-array technology. The MAS is built on top of the Bioconductor which is an Open Source Software for bio-informatics, a platform in which a wide range of statistical and graphical methods for the analysis of high-throughput genomic data can be incorporated. It is a prototype system only. Implementation and validation of the system still need to be done and more machine learning algorithms required.

In [117] authors proposed a prototype MAS implemented using Java Agent DEvelop-

ment Framework (JADE) [20] for biological data analysis. It is a prototype system only. Implementation and validation of the system still need to be done.

In [63] authors proposed the use of agent technology for designing bio-informatics integration systems and in [38] authors presented the development of *BioMen* (Biological Management Executed over Network) managed by means of a MAS. Authors in [19] discussed the use of DM and MAS for genome annotation pipeline process and in [46] proposed a MAS for remote data retrieval for bio-informatics applications and discussed the synchronous and asynchronous migration strategies for agents. Authors in [28] proposed an Android based MAS for remote medical monitoring of diabetic patients. The main aim is for the improvement of the transmission of information between patients and their physicians, especially the management of specific and critical cases. Recently authors in [79] designed a multi-agent system for gene expression analysis (MAS-GEN). A group of specialized agents with different abilities handle distribution of activities to reduce the time and complexity of the analysis.

2.2.2 Discussions

Sequencing of organisms results into generation of huge quantity of dynamic, unstructured, heterogeneously distributed and exponentially growing raw biological data. Sharing this high-throughput biological data and making sense (or mining interesting knowledge) out of this data is a common problem among every one involved in this field [17, 24, 89, 117]. The lack of network bandwidth is also one of the main limitation for user activities which prohibits efficient transmission and receiving of huge biological data [82]. The post-genomic era has resulted into availability of the enormous amount of distributed biological data sets that require suitable tools and methods for modeling and analyzing biological processes and sequences. The bio-informatics research community feels a strong need to develop new models and exploit and analyze the available genomes [41].

It is observed that most of the MAS in bio-informatics domain are prototype based academic research projects designed for a specific biological problem with no implementation, validation and verification. Hermes is the only AEE acting as middleware specifically designed for WMS for biological domain in a heterogeneous environment. Genome analysis and annotation, Protein structure prediction, modeling and simulation of biological process, workflow management system are the main areas in bio-informatics where the software agent technology is being currently used. DECAF [49] and JADE [20] are the prominent AEE used for designing a MAS. Java, Perl, Prolog and C++ are the main programming languages in use for implementing the MAS. Lots of work is still required for actually implementing such MAS in a distributed environment using real biological data sets. In [51], authors mine the association rules among amino acids in protein sequences. As per the literature available, no researcher has applied the agent technology to mine association rules for amino acids from distributed protein data sets.

2.3 Summary

This chapter presents a review of the current research relating to Agent based DARM and MAS in bio-informatics domain. It also discusses the issues involved in agent based

DARM task and biological data mining. In the next chapter, a central DW based approach for DM is discussed.

Chapter 3

DATA WAREHOUSE BASED ASSOCIATION RULE MINING

3.1 Introduction

The traditional approach for knowledge discovery technique in distributed environment involves the creation of a single centrally integrated data repository called Data Warehouse (DW). DM techniques are used to mine this central DW and extract the knowledge [91]. However, this approach is ineffective because of heavy storage and computational costs involved in managing data from the ever increasing and updated distributed resources where data is produced continuously in streams. Network communication cost is also involved while transferring huge data over the wired or wireless network in a limited network bandwidth scenario. It is also not desirable to centrally collect the privacy-sensitive raw distributed data of the business organizations like banking, and telecommunication as they want only knowledge to be exchanged globally. Data from modern business organizations are not only geographically distributed but also horizontally or vertically fragmented making it difficult if not possible to combine them in a central location. Performance and scalability of a DM application can be increased by distributing the workload among sites [109]. Distributed and mobile environment has certain resource constraints which are not considered in algorithms for central DW based DM as certain data sets are immovable in practice.

Rest of the chapter is organised as follows. Section 3.2 describes the tool used for generating synthetic datasets for ARM process. Central DW based ARM model is described in Section 3.3. It discusses the preliminaries notations, datasets used, framework layout and various algorithms involved in the framework. Overall time model for ARM task is also presented. Experimental validation and results are shown in Section 3.4. Section 3.5 highlights the issues involved in this approach and finally the chapter is summarized in Section 3.6.

3.2 Synthetic Dataset Generation

A transactional dataset is required for mining the ARs from the frequent itemsets. This dataset can be a real dataset of any retail industry or can be a synthetic version generated by a tool. This dataset is used to for experimenting with the DM concepts and mining the

ARs from the frequent itemsets. Such synthetic datasets are used to test and experiment the newly designed algorithms and the concepts can be implemented on a real dataset thereafter. For generating the synthetic dataset a tool called *Transactional Data Set Generator (TDSG)* is developed.

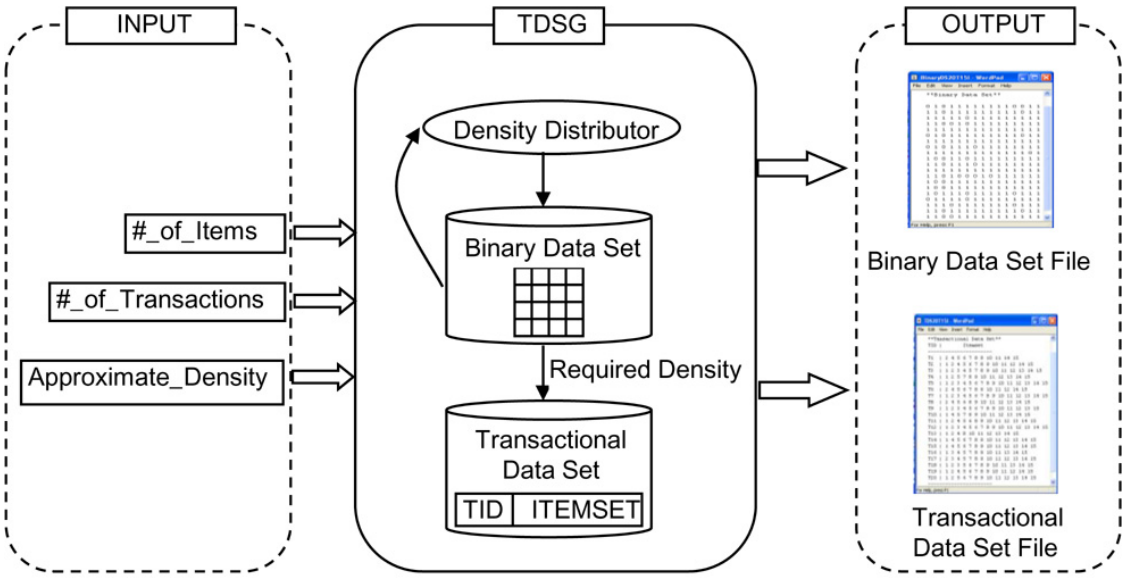

Figure 3.1: Block Diagram of TDSG

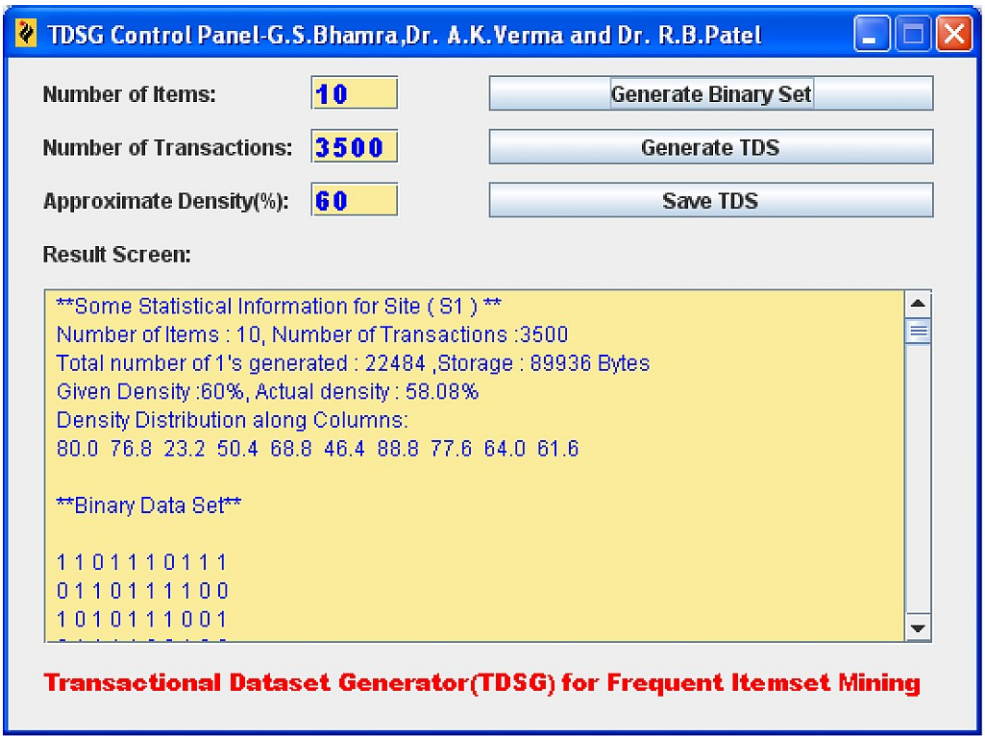

Figure 3.2: Control Panel of TDSG

The block diagram of *TDSG* is shown in Fig. 3.1. Binary dataset (BDS) as well as Transactional dataset (TDS) are generated by this tool. *TDSG* takes total number of items in a transactional dataset, total number of transactions and approximate density of the number of 1s in BDS as inputs. A 2-D array of binary values is created with number of columns equals number of items and number of rows equals number of transactions. Binary value 1 in the array represents a purchased item and 0 otherwise. Density Distributor component repeated attempts to get the desired density of array. When a BDS of required density is created then a TDS is generated. Each transaction in TDS consists of Transaction Identification (TID) and all the items purchased in that transaction. *TDSG* tool generates two text files as output, a BDS text file (e.g. BinaryDS20T10I.txt) for BDS and TDS text file (e.g. TDS20T10I.txt) for TDS. These files are stored at desired location in the local file system.

TDSG is implemented in Java language. A maximum of 2^{32} number of items and 2^{32} number of transactions in a data set are supported by this tool. Thus, it generates a Binary datasets of 2^{64} items. Control panel of *TDSG* is shown in Fig. 3.2.

3.3 Central DW based ARM

3.3.1 Preliminaries and Definitions

A client-server based framework for mining strong association rules from central DW uses the following terms:

- $S = \{S_i, i = 1 \dots n\}$, n distributed sites
- $S_{CENTRAL}$, a Central site
- $DB_i = \{T_j, j = 1 \dots D_i\}$, Transactional dataset of size D_i at the local site S_i, where each transaction T_j is assigned an identifier (TID)
- $I = \{d_l, l = 1 \dots m\}$, total m data items in each DB_i
- $DB = \bigcup_{i=1}^{n} DB_i$, the Central Data Warehouse at $S_{CENTRAL}$ of size $D = \sum_{i=1}^{n} D_i$
- min_th_sup, given minimum threshold support
- min_th_conf, given minimum threshold confidence
- L^{FI}, the list of all frequent k-itemsets at $S_{CENTRAL}$
- L^{FISC}, the list of support count of every frequent k-itemsets in L^{FI}
- L^{SAR}, the list of strong association rules at$S_{CENTRAL}$

3.3.2 Synthetic Datasets

Synthetic dataset (DB_i) is stored across three distributed sites S_1, S_2 and S_3, with 3500, 3850 and 3900 transactions and 10 items in each respectively using *TDSG* tool. Table 3.1 shows size and density of each DB_i. Binary and transactional versions of these datasets are shown in Appendix A.

Table 3.1: Size and density of each DB_i.

Site Name	No. of Items	No. of Transactions	Approximate Density	Size of Data Set
S1	10	3500	60%	172KB
S2	10	3850	50%	192KB
S3	10	3900	65%	192KB

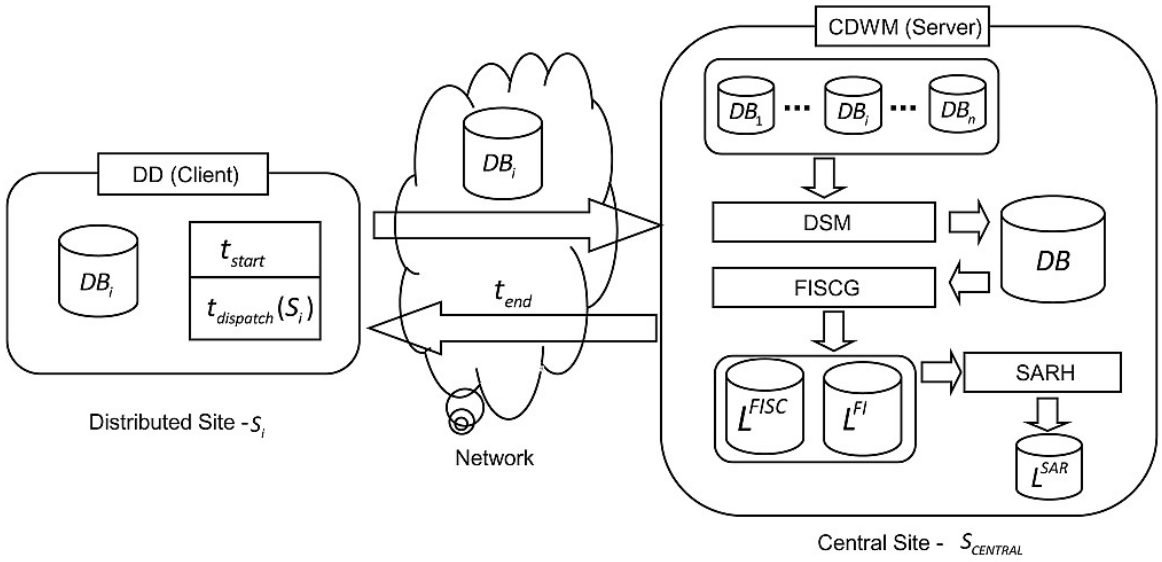

Figure 3.3: Framework for Central DW based ARM

3.3.3 Framework for Central DW based ARM

The client-server model for central DW based ARM approach is depicted in Fig. 3.3. Various components involved in the framework are:

1. ***Dataset Dispatcher (DD)***: It is a client application running at each distributed site S_i to dispatch DB_i over the network to the *Central Data Warehouse Manager (CDWM)*. *CDWM* is a server application running at $S_{CENTRAL}$. *DD* also keeps track of the time taken to dispatch DB_i from i^{th} site, i.e., $t_{dispatch}(S_i)$. It first waits for the network channel (α) to be available and then binds object input (OIS_α) and object output (OOS_α) streams with α. Entire DB_i is retrieved from the local file system using input stream (OIS) and start time(t_{start}) for dispatching the DB_i is maintained at this site S_i. DB_i is then transferred to *CDWM* over OOS_α stream. It

finally retrieves end time(t_{end}) using OIS_α stream and calculate $t_{dispatch}(S_i)$. Steps involved in the process is given in Algorithm 1.

In general, $t_{dispatch}(S_i)$ depends upon the following factors: time taken in establishing a socket connection with *CDWM* at $S_{CENTRAL}$ ($t_{connection}(S_{CENTRAL})$), size and density of the dataset ($Size\_Density(DB_i)$) to be dispatched and the network bandwidth (in Kbits per second) between two nodes S_i and $S_{CENTRAL}$ ($BWidth(S_i, S_{CENTRAL})$). Therefore,

$$t_{dispatch}(S_i) \propto \frac{k \times t_{connection}(S_{CENTRAL}) \times Size\_Density(DB_i)}{BWidth(S_i, S_{CENTRAL})} \tag{3.1}$$

where k is a constant.

2. ***Central Data Warehouse Manager (CDWM)***- is a multi-threaded server application running at $S_{CENTRAL}$. It handles all the incoming requests from *DD*. *CDWM* receives & stores each DB_i and waits continuously till all the clients have dispatched their DB_i. For each incoming request from DD, it first establishes a new socket (β) with DD and starts a new thread of *CDWMImpl* module to handle that request. *CDWMImpl* module binds input stream (OIS_β) and output stream (OOS_β) with channel β and retrieves entire DB_i object at OIS_β and stores the same in local file system at server and it also sends the data received time (t_{end}) back to the DD using OOS_β stream.

   Let $t_{connection}(S_i)$ be the time taken in establishing a socket connection with *DD* at S_i. Therefore, total waiting time (t_{wait}) is calculated as follows-

$$t_{wait} = \sum_{i=1}^{n} t_{connection}(S_i) \tag{3.2}$$

   Detailed steps are shown in Algorithm 2. Various other assistant components of *CDWM* application are:-

   (a) ***Dataset Merger (DSM)*** merges all the collected set of DB_i from distributed sites into a single integrated unit ($DB = \bigcup_{i=1}^{n} DB_i$). Steps involved in the process are shown in Algorithm 3.

   Let $t_{merging}$ be the time taken in merging all the DB_i and it depends on the following factors: $Size\_Density(DB_i)$, no. of other processes running at the central site ($\#\_of\_Processes\,(S_{CENTRAL})$) and the processor speed at $S_{CENTRAL}$ ($Processor\_Speed\,(S_{CENTRAL})$). Therefore,

$$t_{merging} \propto \frac{k \times Size\_Density(DB_i) \times \#\_of\_Processes(S_{CENTRAL})}{Processor\_Speed(S_{CENTRAL})} \tag{3.3}$$

   where, k is a constant.

   (b) ***Frequent Itemsets and Support Counts Generator (FISCG)***: It implements Apriori [5] algorithm to scan merged dataset *DB* with *min_th_sup* and generates list of frequent itemsets (L^{FI}) and list of support counts for frequent itemsets (L^{FISC}). Detailed steps are shown in Algorithm 4. It makes use

of a module described in Algorithm 5 to generate candidate frequent k-itemset list by joining a frequent (k-1)-itemset with itself. Subset testing is performed by another module described in Algorithm 6.

Let t_{fim} be the time taken in frequent itemset mining of merged dataset (DB). It depends upon the following factors: $Size\_Density(DB)$, time complexity (t_{fim_algo}) of underlying FIM algorithm, $\#\_of\_Processes\,(S_{CENTRAL})$ and $Processor\_Speed\,(S_{CENTRAL})$. Therefore,

$$t_{fim} \propto \frac{k \times Size\_Density(DB) \times t_{fim\_algo} \times \#\_of\_Processes(S_{CENTRAL})}{Processor\_Speed(S_{CENTRAL})} \tag{3.4}$$

where, k is a constant.

(c) ***Strong Association Rule Harvester (SARH)***: It generates the list of strong association rules (L^{SAR}) from L^{FI} and L^{FISC} with the constraint of min_th_conf for the total number of records (T). $SARH$ iteratively retrieves list of frequent k-itemsets (L_k) from L^{FI} for k=2 to the size of L^{FI} and for all itemsets ($l \in L_k$), it generates all non-empty subsets ($l_{subsets}$) of l. Support count of l, i.e., $l_{supcount}$ is retrieved from L^{FISC}. It computes support of the desired association rule ($AR_{support}$) using the formula $AR_{support} = l_{supcount}/T \times 100$. For all non-empty subset ($s \in l_{subsets}$), support count of s, i.e., $s_{supcount}$ is retrieved from L^{FISC} and confidence of the desired association rule (AR_{conf}) is calculated using the formula $AR_{conf} = l_{supcount}/s_{supcount} \times 100$. If $AR_{conf} \geq min\_th\_conf$ then strong association rule (AR_{strong}) is generated as "$s \Rightarrow l - s[AR_{support}\%, AR_{conf}\%]$" and this AR_{strong} is added to L^{SAR}. Steps used by $SARH$ are given in Algorithm 7.

Let t_{arm} be the time taken in association rule mining from L^{FI}. It depends upon the following factors: the size of L^{FI} list ($Size\left(L^{FI}\right)$), $\#\_of\_Processes\,(S_{CENTRAL})$ and $Processor\_Speed\,(S_{CENTRAL})$. Therefore,

$$t_{arm} \propto \frac{k \times Size(L^{FI}) \times \#\_of\_Processes(S_{CENTRAL})}{Processor\_Speed(S_{CENTRAL})} \tag{3.5}$$

where, k is a constant.

3.3.4 Overall Time Model for Central DW based ARM

Using equations 3.1 (longest time taken), 3.2, 3.3, 3.4 and 3.5, the overall response time(T) for the ARM task performed using Central DW based framework is calculated using the following equation:

$$T = max(t_{dispatch}(S_i)) + t_{wait} + t_{merging} + t_{fim} + t_{arm} \tag{3.6}$$

Algorithm 1 DATASET DISPATCHER (DD)

Input: $DB_i, Transactional\ Data\ Set\ at\ site\ S_i$
Output: $t_{dispatch}(S_i), Total\ time\ taken\ in\ dispatching\ DB_i$

1: **procedure** DD(DB_i)
2: $\alpha \leftarrow$ *establish a socket channel with* $CDWM$
3: **if** α *channel is established* **then**
4: $OIS_\alpha \leftarrow$ *bind ObjectInputStream at* α
5: $OOS_\alpha \leftarrow$ *bind ObjectOutputStream at* α
6: $OIS \leftarrow$ *bind ObjectInputStream with local* DB_i
7: $DB_i \leftarrow$ *read the entire* DB_i *object at* OIS *stream*
8: $t_{start} \leftarrow$ *get start time for dataset transfer*
9: *dispatch* DB_i *on* OOS_α *stream*
10: $t_{end} \leftarrow$ *read end time received on* OIS_α *stream*
11: $t_{dispatch}(S_i) \leftarrow t_{end} - t_{start}$
12: *Close all the opened channels*
13: **return** $t_{dispatch}(S_i)$
14: **else**
15: *Go to step* 2
16: **end if**
17: **end procedure**

Algorithm 2 CENTRAL DATA WAREHOUSE MANAGER (CDWM)

Input:

- $DB = \{DB_i, i = 1 \cdots n\}, Transactional\ Data\ Set\ from\ n\ distributed\ sites$
- $I = \{d_l, l = 1 \cdots m\}, total\ m\ items\ in\ DB_i$
- $min\_th\_sup, the\ given\ minimum\ threshold\ support$
- $min\_th\_conf, the\ given\ minimum\ threshold\ confidence$

Output: $L^{SAR}, the\ list\ of\ strong\ association\ rules$

1: **procedure** CDWM($DB, I, min\_th\_sup, min\_th\_conf$)
2: $P \leftarrow$ *open a new server port at* # 9900
3: **while** *all* DB_i *from registered clients not received* **do**
4: $\beta \leftarrow$ *accept the incoming request at* P *and open a new socket with the client*
5: *start a new Thread of* CDWMIMPL(β)
6: **end while**
7: $DB \leftarrow$ *Call* DSM($\{DB_i, i = 1 \cdots n\}$) ▷ see Algorithm 3
8: $L^{FI\&SC} \leftarrow$ *Call* FISCG(DB, I, min_th_sup) ▷ see Algorithm 4
9: $L^{SAR} \leftarrow$ *Call* SARH($L^{FI\&SC}, min\_th\_conf$) ▷ see Algorithm 7
10: *Close all the opened channels*
11: **return** L^{SAR}
12: **end procedure** ▷ continue...

CDWM - continued

13: **procedure** CDWMIMPL(β : *socket with the client*)
14: $OIS_\beta \leftarrow$ *bind ObjectInputStream at* β
15: $OOS_\beta \leftarrow$ *bind ObjectOutputStream at* β
16: $DB_i \leftarrow$ *read the entire* DB_i *object at* OIS_β *stream*
17: $t_{end} \leftarrow$ *get end time for dataset receiving*
18: *write* t_{end} *on* OOS_β *stream to send to the client*
19: $OOS \leftarrow$ *bind ObjectOutputStream with local file system on server*
20: *write* DB_i *on* OOS *stream to save it on the server*
21: *Close all the opened channels*
22: **end procedure**

Algorithm 3 DATASET MERGER (DSM)

Input: $\{DB_i, i = 1 \cdots n\}$
Output: $DB, AggregatedDataset$

1: **procedure** DSM($\{DB_i, i = 1 \cdots n\}$)
2: **for all** $DS \in \{DB_i, i = 1 \cdots n\}$ **do**
3: $OIS_{DS} \leftarrow$ *bind ObjectInputStream with DS*
4: $DB_i \leftarrow$ *read the entire* DB_i *object at* OIS_{DS} *stream*
5: *add* DB_i *into a single unit DB* ▷ $DB = \bigcup_{i=1}^{n} DB_i$
6: **end for**
7: $OOS \leftarrow$ *bind ObjectOutputStream with local file system on server*
8: *write merged DB on OOS stream to save it on the server*
9: *Close all the opened channels*
10: **return** DB
11: **end procedure**

Algorithm 4 FREQUENT ITEMSETS AND SUPPORT COUNTS GENERATOR (FISCG)

Input:

- $DB = \bigcup_{i=1}^{n} DB_i, Central\ Datawarehouse$
- $I = \{d_l, l = 1 \cdots m\}, total\ m\ items\ in\ DB_i$
- $min\_th\_sup, the\ given\ minimum\ threshold\ support$

Output: $L^{FI\&SC}, the\ list\ of\ frequent\ itemsets\ their\ support\ counts$

1: **procedure** FISCG($DB, Itemset, min\_th\_sup$)
2: $T \leftarrow DB.size$ ▷ No. of records in DB
3: $I \leftarrow Itemset.size$ ▷ No. of items in DB
4: $min\_sup\_count \leftarrow (T \times min\_th\_sup)/100$
5: ▷ generate frequent-1 itemset list (FIL_1) and support count list ($FISC_1$)
6: $CFIL_1 \leftarrow \{1, 2, 3 \cdots I\}$ ▷ candidate frequent-1 itemset
7: **for** $i \leftarrow 1, I$ **do** ▷ initialize the support count array $SCFIL_1$ to zero
8: $SCFIL_1[i] \leftarrow 0$ ▷ continue...

FISCG - continued

```
    end for
    k ← 1
    for all candidate c ∈ CFIL_1 do                ▷ find support count for every candidate
        for all transaction t ∈ DB do                                          ▷ scan DB
            if c ⊂ t then
                SCFIL_1[k] ← SCFIL_1[k] + 1
            end if
        end for
        k ← k + 1
    end for
    ▷ prune CFIL_1 to generate FIL_1 and FISC_1
    for k ← 1, I do
        if SCFIL_1[k] ≥ min_sup_count then
            add c_k ∈ CFIL_1 to FIL_1
            add SCFIL_1[k] to FISC_1
        end if
    end for
    if FIL_1 ≠ ∅ then
        add FIL_1 to L^FI
        add FISC_1 to L^FISC
    end if
    k ← 2
    while FIL_{k-1} ≠ ∅ do
        CFIL_k ← Call GenerateCFIL(FIL_{k-1})                          ▷ see Algorithm 5
        for i ← 1, CFIL_k.length do          ▷ initialize the array SCFIL_k to zero
            SCFIL_k[i] ← 0
        end for
        i ← 1
        for all candidate c ∈ CFIL_k do            ▷ find support count for every candidate
            for all transaction t ∈ DB do                                      ▷ scan DB
                if c ⊂ t then
                    SCFIL_k[i] ← SCFIL_k[i] + 1
                end if
            end for
            i ← i + 1
        end for
        ▷ prune CFIL_k to generate FIL_k and FISC_k
        for i ← 1, SCFIL_k.length do
            if SCFIL_k[i] ≥ min_sup_count then
                add c_i ∈ CFIL_k to FIL_k
                add SCFIL_k[i] to FISC_k
            end if
        end for
        if FIL_k ≠ ∅ then
            add FIL_k to L^FI                                          ▷ continue...
```

FISCG - continued

```
54:             add FISC_k to L^FISC
55:         end if
56:         k ← k + 1
57:     end while
58:     add T to L^FI&SC
59:     add L^FI to L^FI&SC
60:     add L^FISC to L^FI&SC
61:     return L^FI&SC
62: end procedure
```

Algorithm 5 GENERATECFIL

Input: $L_{k-1}, Frequent\ k-1\ itemsets$
Output: $C_k, Candidate\ Frequent\ k\ itemsets$

```
1: procedure GENERATECFIL(L_{k-1})
2:     for all itemset l_1 ∈ L_{k-1} do
3:         for all itemset l_2 ∈ L_{k-1} do
4:             if (l_1[1] = l_2[1]) ∧ (l_1[2] = l_2[2]) ∧ ··· ∧ (l_1[k-1] = l_2[k-1]) then
5:                 c ← l_1 ⊗ l_2                                   ▷ join step
6:             end if
7:             if HASINFREQUENTSUBSET(c, L_{k-1}) then             ▷ see Algorithm 6
8:                 delete c                                        ▷ prune step
9:             else
10:                add c to C_k
11:            end if
12:        end for
13:    end for
14:    return C_k
15: end procedure
```

Algorithm 6 HASINFREQUENTSUBSET

Input: $c, Candidate\ k-itemset$
Output: $L_{k-1}, Frequent\ k-1\ itemsets$

```
1: procedure HASINFREQUENTSUBSET(c, L_{k-1})
2:     for all (k-1) subset s ∈ c do
3:         if s ∉ L_{k-1} then
4:             return TRUE
5:         else
6:             return FALSE
7:         end if
8:     end for
9: end procedure
```

Algorithm 7 STRONG ASSOCIATION RULE HARVESTER (SARH)

Input:

- $L^{FI\&SC}$, *Collection of number of records , frequent k itemset & support count*
- min_th_conf, *the given minimum threshold confidence*

Output: L^{SAR}, *List of Strong Association Rules*

procedure SARH($L^{FI\&SC}, min\_th\_conf$)
  $T \leftarrow L^{FI\&SC}.get(0)$ ▷ No. of records
  $L^{FI} \leftarrow L^{FI\&SC}.get(1)$ ▷ frequent k-itemset list
  $L^{FISC} \leftarrow L^{FI\&SC}.get(2)$ ▷ support count list
  for $k \leftarrow 2, L^{FI}.size$ **do**
    $L_k \leftarrow L^{FI}.get(k)$ ▷ get frequent k-itemset list
    for all $l \in L_k$ **do**
      $l_{subsets} \leftarrow generate\ all\ non-empty\ subsets\ of\ l$
      $l_{spcount} \leftarrow get\ support\ count\ of\ l\ from\ L^{FISC}$
      $AR_{support} \leftarrow (l_{spcount}/T) \times 100$ ▷ support of the association rule
      for all $non-empty\ subset\ s \in l_{subsets}$ **do**
        $s_{spcount} \leftarrow get\ support\ count\ of\ s\ from\ L^{FISC}$
        $AR_{conf} \leftarrow (l_{spcount}/s_{spcount}) \times 100$ ▷ confidence of the association rule
        if $AR_{conf} \geq min\_th\_conf$ **then**
          $AR_{strong} \leftarrow$ "$s \Rightarrow l - s[AR_{support}\%, AR_{conf}\%]$"
          print AR_{strong}
          $add\ AR_{strong}\ to\ L^{SAR}$
        end if
      end for
    end for
  end for
  return L^{SAR}
end procedure

3.4 Implementation and Performance Study

All the components of the framework are implemented in Java language. Table 3.2 describes required network configuration for the framework. Fig. 3.4 shows the snapshots of the running state of the *DD* at each distributed site. Time taken by each *DD* in transferring DB_i over the network is shown in Fig. 3.5. Fig. 3.6 shows the snapshot of the running state of the *CDWM* at $S_{CENTRAL}$. L^{FI} and L^{FISC} lists generated by *FISCG* module with 20% min_th_sup is shown in Fig. 3.7.

Table 3.2: Network configuration.

Site Name	Processor	OS	LAN Configuration	
			IP^a	Network
$S_{CENTRAL}$	$Intel^b$	MS^c	192.168.46.5	NW^d
S_1	$Intel^b$	MS^c	192.168.46.212	NW^d
S_2	$Intel^b$	MS^c	192.168.46.189	NW^d
S_3	$Intel^b$	MS^c	192.168.46.213	NW^d

a. IP address with Mask:255.255.255.0 and Gateway:192.168.46.1

b. Intel Pentium Dual Core(3.40 GHz,3.40 GHz)with 512MB RAM

c. Microsoft Windows XP Professional ver. 2002

d. Network Speed:100 Mbps and Network Adaptor: Intell 82566DM-2 Gigabit NIC

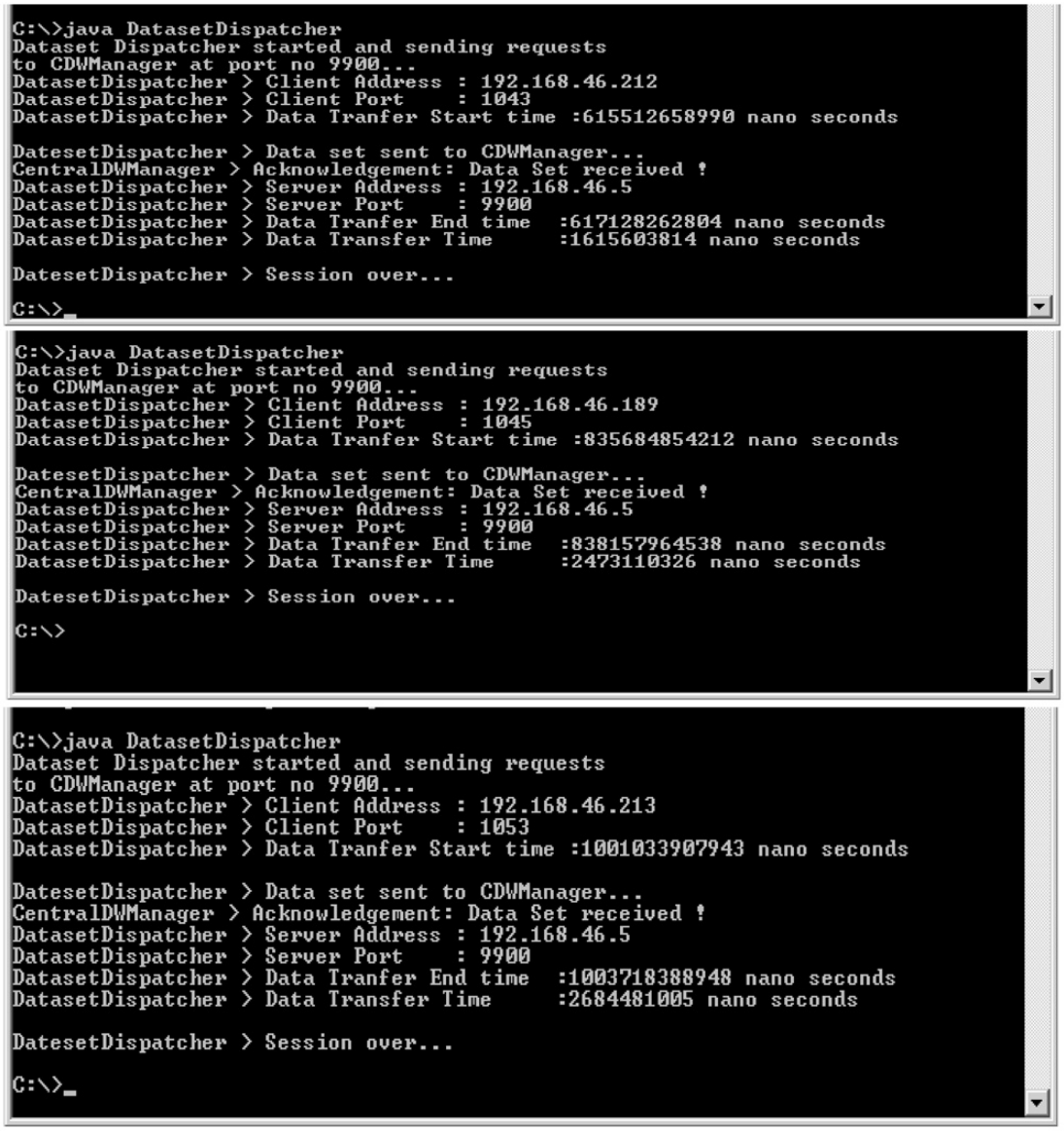

Figure 3.4: Dataset Dispatcher running at S_1, S_2 and S_3

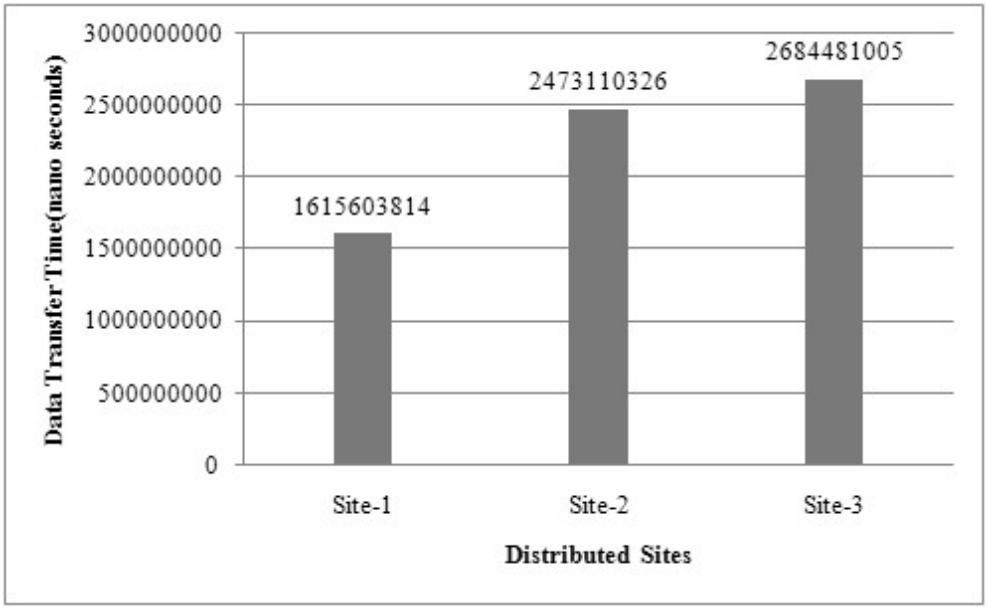

Figure 3.5: Dataset Transfer Time ($t_{dispatch}$) at S_1, S_2 and S_3

```
C:\WINNT\system32\cmd.exe - java CentralDWManager

C:\>java CentralDWManager
CentralDWManager > Central Data Warehouse Manager started....
CentralDWManager > Server Address : 192.168.46.5
CentralDWManager > Server Port    : 9900

CentralDWManager > Client Request Number: 1
CentralDWManager > Client Address : 192.168.46.212
CentralDWManager > Client Port    : 1043
CentralDWManager > Data receiving Time : 617128262804 nano seconds
CentralDWManager > Data set received from Site : S1
CentralDWManager > Number of Transactions      : 3500
CentralDWManager > Number of Items             : 10
CentralDWManager > Data set saved at c:\Pmade1.2\CentralDW with name : S1.dat
CentralDWManager > Acknowledgement sent ...

CentralDWManager > Client Request Number: 2
CentralDWManager > Client Address : 192.168.46.189
CentralDWManager > Client Port    : 1045
CentralDWManager > Data receiving Time : 838157964538 nano seconds
CentralDWManager > Data set received from Site : S2
CentralDWManager > Number of Transactions      : 3850
CentralDWManager > Number of Items             : 10
CentralDWManager > Data set saved at c:\Pmade1.2\CentralDW with name : S2.dat
CentralDWManager > Acknowledgement sent ...

CentralDWManager > Client Request Number: 3
CentralDWManager > Client Address : 192.168.46.213
CentralDWManager > Client Port    : 1053
CentralDWManager > Data receiving Time : 1003718388948 nano seconds
CentralDWManager > Data set received from Site : S3
CentralDWManager > Number of Transactions      : 3900
CentralDWManager > Number of Items             : 10
CentralDWManager > Data set saved at c:\Pmade1.2\CentralDW with name : S3.dat
CentralDWManager > Acknowledgement sent ...

CentralDWManager > All registered clients have submitted their Data sets...

DatasetMerger > DatasetMerger started....
DatasetMerger > Start Time : 1003743382243 nano seconds
DatasetMerger > Dataset name:S1.dat
DatasetMerger > TDS from S1.dat loaded...
DatasetMerger > Dataset name:S2.dat
DatasetMerger > TDS from S2.dat loaded...
DatasetMerger > Dataset name:S3.dat
DatasetMerger > TDS from S3.dat loaded...
DatasetMerger > Total Transactions from all TDS : 11250
DatasetMerger > DB.dat saved...
DatasetMerger > End Time : 1003947171920 nano seconds
DatasetMerger > CPU Time Consumed : 203789677 nano seconds
```

Figure 3.6: Central DW Manager running at $S_{CENTRAL}$

1-Itemsets		
S.n.	L	SC
1	[1]	8511
2	[2]	5672
3	[3]	5387
4	[4]	7277
5	[5]	2509
6	[6]	6606
7	[7]	6973
8	[8]	8781
9	[9]	7993
10	[10]	5087

2-Itemsets		
S.n.	L	SC
1	[1, 2]	4143
2	[1, 3]	3946
3	[1, 4]	5356
4	[1, 6]	5213
5	[1, 7]	5451
6	[1, 8]	6128
7	[1, 9]	6277
8	[1, 10]	3606
9	[2, 3]	2820
10	[2, 4]	3884
11	[2, 6]	3900
12	[2, 7]	3362
13	[2, 8]	4264
14	[2, 9]	4123
15	[2, 10]	2953
16	[3, 4]	3561
17	[3, 6]	3086
18	[3, 7]	3261
19	[3, 8]	4405
20	[3, 9]	3731
21	[3, 10]	2494
22	[4, 6]	4413
23	[4, 7]	4438
24	[4, 8]	5733
25	[4, 9]	5171
26	[4, 10]	3496
27	[6, 7]	4104
28	[6, 8]	4372
29	[6, 9]	5051
30	[6, 10]	3238
31	[7, 8]	5236
32	[7, 9]	5055
33	[7, 10]	3001
34	[8, 9]	5761
35	[8, 10]	4054
36	[9, 10]	3572

3-Itemsets		
S.n.	L	SC
1	[1, 2, 4]	2759
2	[1, 2, 6]	3008
3	[1, 2, 7]	2585
4	[1, 2, 8]	2783
5	[1, 2, 9]	3169
6	[1, 3, 4]	2535
7	[1, 3, 6]	2333
8	[1, 3, 7]	2476
9	[1, 3, 8]	3002
10	[1, 3, 9]	2826
11	[1, 4, 6]	3409
12	[1, 4, 7]	3404
13	[1, 4, 8]	3870
14	[1, 4, 9]	3969
15	[1, 4, 10]	2424
16	[1, 6, 7]	3374
17	[1, 6, 8]	3040
18	[1, 6, 9]	4202
19	[1, 6, 10]	2394
20	[1, 7, 8]	3773
21	[1, 7, 9]	4086
22	[1, 8, 9]	4113
23	[1, 8, 10]	2622
24	[1, 9, 10]	2671
25	[2, 4, 6]	2685
26	[2, 4, 7]	2250
27	[2, 4, 8]	2990
28	[2, 4, 9]	2791
29	[2, 6, 7]	2362
30	[2, 6, 8]	2598
31	[2, 6, 9]	2999
32	[2, 7, 8]	2380
33	[2, 7, 9]	2519
34	[2, 8, 9]	2836
35	[2, 8, 10]	2363
36	[3, 4, 8]	2949
37	[3, 4, 9]	2463
38	[3, 6, 9]	2290
39	[3, 7, 8]	2567
40	[3, 7, 9]	2289
41	[3, 8, 9]	2841
42	[4, 6, 7]	2706
43	[4, 6, 8]	3009
44	[4, 6, 9]	3335
45	[4, 6, 10]	2261
46	[4, 7, 8]	3366
47	[4, 7, 9]	3220
48	[4, 8, 9]	3760
49	[4, 8, 10]	2831
50	[4, 9, 10]	2430
51	[6, 7, 8]	2525
52	[6, 7, 9]	3218
53	[6, 8, 9]	3008
54	[6, 8, 10]	2306
55	[6, 9, 10]	2403
56	[7, 8, 9]	3499
57	[7, 8, 10]	2299
58	[8, 9, 10]	2648

4-Itemsets		
S.n.	L	SC
1	[1, 2, 6, 9]	2457
2	[1, 4, 6, 9]	2728
3	[1, 4, 7, 8]	2371
4	[1, 4, 7, 9]	2569
5	[1, 4, 8, 9]	2602
6	[1, 6, 7, 9]	2774
7	[1, 7, 8, 9]	2575
8	[4, 7, 8, 9]	2252

Figure 3.7: Lists of Frequent Itemsets and their support count with 20% *min_th_sup* at $S_{CENTRAL}$

Stong Association Rules for 4-Itemsets		
S.n.	**L**	**AR (support,confidence)**
1	[1, 2, 6, 9]	[1, 2] => [6, 9] (21%, 59%) [2, 6] => [1, 9] (21%, 63%) [2, 9] => [1, 6] (21%, 59%) [1, 2, 6] => [9] (21%, 81%) [1, 2, 9] => [6] (21%, 77%) [1, 6, 9] => [2] (21%, 58%) [2, 6, 9] => [1] (21%, 81%)
2	[1, 4, 6, 9]	[1, 4] => [6, 9] (24%, 50%) [1, 6] => [4, 9] (24%, 52%) [4, 6] => [1, 9] (24%, 61%) [4, 9] => [1, 6] (24%, 52%) [6, 9] => [1, 4] (24%, 54%) [1, 4, 6] => [9] (24%, 80%) [1, 4, 9] => [6] (24%, 68%) [1, 6, 9] => [4] (24%, 64%) [4, 6, 9] => [1] (24%, 81%)
3	[1, 4, 7, 8]	[4, 7] => [1, 8] (21%, 53%) [1, 4, 7] => [8] (21%, 69%) [1, 4, 8] => [7] (21%, 61%) [1, 7, 8] => [4] (21%, 62%) [4, 7, 8] => [1] (21%, 70%)
4	[1, 4, 7, 9]	[4, 7] => [1, 9] (22%, 57%) [7, 9] => [1, 4] (22%, 50%) [1, 4, 7] => [9] (22%, 75%) [1, 4, 9] => [7] (22%, 64%) [1, 7, 9] => [4] (22%, 62%) [4, 7, 9] => [1] (22%, 79%)
5	[1, 4, 8, 9]	[4, 9] => [1, 8] (23%, 50%) [1, 4, 8] => [9] (23%, 67%) [1, 4, 9] => [8] (23%, 65%) [1, 8, 9] => [4] (23%, 63%) [4, 8, 9] => [1] (23%, 69%)
6	[1, 6, 7, 9]	[1, 6] => [7, 9] (24%, 53%) [1, 7] => [6, 9] (24%, 50%) [6, 7] => [1, 9] (24%, 67%) [6, 9] => [1, 7] (24%, 54%) [7, 9] => [1, 6] (24%, 54%) [1, 6, 7] => [9] (24%, 82%) [1, 6, 9] => [7] (24%, 66%) [1, 7, 9] => [6] (24%, 67%) [6, 7, 9] => [1] (24%, 86%)
7	[1, 7, 8, 9]	[7, 9] => [1, 8] (22%, 50%) [1, 7, 8] => [9] (22%, 68%) [1, 7, 9] => [8] (22%, 63%) [1, 8, 9] => [7] (22%, 62%) [7, 8, 9] => [1] (22%, 73%)
8	[4, 7, 8, 9]	[4, 7] => [8, 9] (20%, 50%) [4, 7, 8] => [9] (20%, 66%) [4, 7, 9] => [8] (20%, 69%) [4, 8, 9] => [7] (20%, 59%) [7, 8, 9] => [4] (20%, 64%)

Figure 3.8: Strong Association Rules for Frequent 4-Itemsets at $S_{CENTRAL}$

Table 3.3: Analysis of the Central DW based ARM approach.

Parameter	Analysis
Storage Cost(cost incurred in storing and managing the data)	Size of central DB = 565248 bytes (552KB) As all data from distributed sites are centrally collected so storage cost is high at central location.
Communication Cost(cost incurred in terms of time taken in transferring the data over the network)	$max(t_{dispatch}(S_i))$ = 2684481005 ns. As entire DB_i is to be transferred to central location for mining so communication cost is also high particularly in case of limited network bandwidth.
Computational Cost(cost incurred in terms of CPU time taken in executing the task)	CPU time taken by $FISCG$ = 291567951788 ns, CPU time taken by $SARH$ = 218382486678 ns. Total CPU time taken by two major modules in Central DW based ARM is 509950438466 nano sec as huge dataset has to be processed at central location.
Global Knowledge	Central DW based ARM does not reflect the global knowledge as it also contains the strong rules for frequent itemsets which may not be locally frequent.
Security	No security in transferring DB_i from each site over the network.
Scalability	Adding more sites would affect the performance of the system by increasing all the costs involved. Hence it is not scalable system.

A total of 51 strong ARs generated by $SARH$ module for frequent 4-itemsets with 50% min_th_conf are shown in Fig. 3.8. CPU time taken by DM, $FISCG$ and $SARH$ modules are 203789677 nano seconds, 291567951788 nano seconds and 218382486678 nano seconds, respectively. The analysis for central DW based ARM approach is shown in Table 3.3.

3.5 Discussion

The practical investigation and analysis indicates that this central DW based DM approach is ineffective because of storage, communication and computational costs involved in managing data. This approach is not privacy-preserving. Scalability is one of the major issue where adding more sites would affect the performance of the system. Performance and scalability of a DM application can be increased by distributing workload among the sites which is possible when the DM is performed locally and only results are carried out at central site for mining global knowledge.

3.6 Summary

In this chapter we have investigated the central DW based approach for ARM. An analysis of the central DW based ARM is also presented. In the next chapter an agent based approach will be discussed for association rule mining of the distributed data to address the scalability issue.

Chapter 4

MULTI AGENT SYSTEM FOR DISTRIBUTED ASSOCIATION RULE MINING

In this chapter a multi agent system (MAS) called Agent enabled Mining of Globally Strong Association Rules (AeMGSAR) is presented. This system comprises of two models as per the mobility pattern. First model is based on the serial mobility and the second one on parallel mobility of agents. In the Serial computing model each MA visits every node in sequence and finally comes back from the last visited node at the central launching station. Further AeMGSAR is enhanced as parallel computing model which is based on the parallel mobility pattern or itinerary of the MA. In this model clones of a MA visits every node in parallel and immediately comes back at the central launching station. The performance of the system is compared with the central data warehouse based approach presented in Chapter 3 as well as other existing systems in the domain.

Rest of the chapter is organised as follows. Section 4.1 described a running environment for the proposed framework along with various algorithms involved. Serial Computing Model of AeMGSAR is presented in 4.2. Overall time models & results are also discussed and a comparative analysis of this model is done with central DW based approach. Section 4.3 explores the Parallel Computing Model of AeMGSAR. Overall time models & results are also discussed. A comparative analysis of AeMGSAR with other existing systems are performed in section 4.4 and finally the chapter is summarized in Section 4.5.

4.1 Environment for the Proposed System

Every MAS needs an underlying AEE to provide a running infrastructure on which agents can be deployed and tested. A running environment has been designed in Java language to execute all the DM agents involved in the proposed models. Various attributes of the MA are encapsulated within a data structure known as $AgentProfile$. $AgentProfile$ contains the name of MA ($AgentName$), version number ($AgentVersion$), entire byte code (BC), list of nodes to be visited by MA, i.e., itinerary plan (L^{NODES}), type of the itinerary ($ItinType$) which can be serial or parallel, a reference of current execution state ($AObject$) and an additional data structure known as $Briefcase$ that acts as a result bag of MA to store final resultant knowledge ($Result\_S_i$) at a partic-

ular site. Computational time ($CPUTime$) taken by a MA at particular site is also stored in $Result\_S_i$. In addition to the results, $Briefcase$ also contains the system time for start of agent journey ($TripTime_{start}$), system time for end of agent journey ($TripTime_{end}$) and total round trip time of MA ($TripTime$) calculated using the formula $TripTime \leftarrow TripTime_{end} - TripTime_{start}$. Stationary as well as mobile agents involved in the models would be discussed later on. This environment consists of the following three components:

- ***Data Mining Agent Execution Environment (DM_AEE)***: It is the key component that acts as a Server. DM_AEE is deployed on any distributed sites (S_i) and is responsible for receiving, executing and migrating all the visiting DM agents. It receives the incoming $AgentProfile$ at site S_i, retrieves the entire BC of agent and save the BC with $AgentName.class$ in the local file system of site S_i after that the execution of MA is started using $AObject$. Steps are shown in Algorithm 8.

- ***Agent Launcher (AL)***: It acts as a Client at agent launching station ($S_{CENTRAL}$) and launches the goal oriented DM agents on behalf of the user through a user interface to the DM_AEE running at the distributed sites. Agent Pool (or Zone) at $S_{CENTRAL}$ is a repository of all the Mobile as well as Stationary agents(SAs). AL first reads and stores $AgentName$ in $AgentProfile$. The entire BC of $AgentName$ is loaded from Agent Pool and stored in $AgentProfile$. L^{NODES} and $ItinType$ are retrieved and stored in $AgentProfile$. $TripTime_{start}$ is maintained in $Briefcase$ which is further added to $AgentProfile$. In case of serial computing model, i.e., if $ItinType = Serial$, AL dispatches a specific single MA along with L^{NODES}, and it travels from node-to-node. $AgentVersion$ is set as 1 for this agent. In case of parallel computing model, i.e., if $ItinType = Parallel$, AL creates the L^{NODES} number of clones of the specific MA and dispatches each clone in parallel to all the sites listed in L^{NODES}. Here L^{NODES} number of thread are created to dispatch the MAs in parallel. Each clone has $AgentVersion$ starting from 1 to size of L^{NODES}, which is used to identify each clone on the network. Before dispatching the clone of a MA to DM_AEE, the current state of the newly created i^{th} clone object ($AObject[i]$) is also stored in $AgentProfile$. AL also contacts the Result Manager (RM) for processing the $Briefcase$ of an agent. Detailed steps are given in Algorithm 9.

- ***Result Manager (RM)***: It manages and processes the $Briefcase$ of all MAs. RM is either contacted by a MA for submitting its results or by AL for processing the results of specific MA. On completion of itinerary, each DM agent submits its results to RM which computes total round trip time ($TripTime$) of that MA and saves it in the $Briefcase$ of that agent. if $ItinType = Serial$ then it saves the updated $AgentProfile$ of an agent at $S_{CENTRAL}$. if $ItinType = Parallel$ then it saves the $AgentProfile$ of all the clones of the agent with $AgentName$ in a collection $L^{AllProfiles}_{AgentName}$. It is assumed that all the clones report their results to RM. RM may be equipped with the feature of handling non reporting clones by issuing an alert to AL for that clone with specific $AgentVersion$. AL then launch a new clone for the specific $AgentVersion$ for the specific site. When it is contacted by AL for processing the results of a specific agent it sends back either $AgentProfile$ of an agent or a collection $L^{AllProfiles}_{AgentName}$ for all the clones depending on the $ItinType$. Steps are defined in Algorithm 10.

Algorithm 8 Data Mining Agent Execution Environment (DM_AEE)

```
1: procedure DM_AEE( )
2:     while TRUE do
3:         AgentProfile ← listen and receive the incoming AgentProfile at site S_i
4:         AgentName ← get AgentName from AgentProfile
5:         BC ← retrieve the BC of the agent from AgentProfile
6:         save the BC with name AgentName.class in the local file system of S_i
7:         AObject ← get AObject from AgentProfile                ▷ current state
8:         AObject.run()                          ▷ start executing mobile agent
9:     end while
10: end procedure
```

Algorithm 9 Agent Launcher (AL)

```
1: procedure AL( )
2:     option ← read option(dispatch/result)
3:     switch option do
4:         case dispatch                  ▷ dispatch the mobile agent to DM_AEE
5:             AgentName ← read Mobile Agent's name
6:             add AgentName to AgentProfile
7:             BC ← load entire byte code of AgentName from AgentPool
8:             add BC to AgentProfile
9:             L^NODES ← read Itinerary(IP addresses) of mobile agent
10:            ItinType ← read ItinType(Serial/Parallel)
11:            add ItinType to AgentProfile
12:            if ItinType = "Serial" then                    ▷ Serial Itinerary
13:                AgentVersion ← 1
14:                add AgentVersion to AgentProfile
15:                add L^NODES to AgentProfile
16:                TripTime_start ← get system time for start of agent journey
17:                add TripTime_start to Briefcase
18:                add Briefcase to AgentProfile
19:                switch AgentName do
20:                    case LFIGA
21:                        minthrsup ← read minimum threshold support
22:                        AObject ← new LFIGA(AgentProfile, minthrsup)
23:                    end case
24:                    case LKGA
25:                        minthrconf ← read minimum threshold confidence
26:                        AObject ← new LKGA(AgentProfile, minthrconf)
27:                    end case
28:                    case TFICA
29:                        AObject ← new TFICA(AgentProfile)
30:                    end case                                ▷ continue...
```

AL - continued

```
31:                 case LKCA
32:                     AObject ← new LKCA(AgentProfile)
33:                 end case
34:                 case GKDA
35:                     L^GSAR_CENTRAL ← load L^GSAR_CENTRAL generated by GKGA at S_CENTRAL
36:                     add L^GSAR_CENTRAL to Briefcase
37:                     add updated Briefcase to AgentProfile
38:                     AObject ← new GKDA(AgentProfile)
39:                 end case
40:             end switch
41:             add AObject to AgentProfile                          ▷ current state
42:             Transfer AgentProfile to DM_AEE at first IP address in L^NODES
43:         end if
44:         if ItinType = "Parallel" then                            ▷ Parallel Itinerary
45:             AObject[L^NODES.size]   ▷ Array of Agent Objects for clone references
46:             TripTime_start ← get system time for start of agent journey
47:             add TripTime_start to Briefcase
48:             add Briefcase to AgentProfile
49:             switch AgantName do
50:                 case LKGA_P
51:                     min_s ← read minimum threshold support
52:                     min_c ← read minimum threshold confidence
53:                     for i ← 1, L^NODES.size do          ▷ for each node in the itinerary
54:                         AgentVersion ← i
55:                         add AgentVersion to AgentProfile
56:                         NodeAddress ← L^NODES.get(i)              ▷ get an IP address
57:                         AObject[i] ← new LKGA_P(AgentProfile, min_s, min_c)
58:                         Add AObject[i] to AgentProfile            ▷ clone's state
59:                         Transfer AgentProfile to DM_AEE at NodeAddress
60:                     end for
61:                 end case
62:                 case LKCA_P
63:                     for i ← 1, L^NODES.size do          ▷ for each node in the itinerary
64:                         AgentVersion ← i
65:                         add AgentVersion to AgentProfile
66:                         NodeAddress ← L^NODES.get(i)              ▷ get an IP address
67:                         AObject[i] ← new LKCA_P(AgentProfile)
68:                         Add AObject[i] to AgentProfile            ▷ clone's state
69:                         Transfer AgentProfile to DM_AEE at NodeAddress
70:                     end for
71:                 end case                                          ▷ continue...
```

AL - continued

```
72:                 case GKDA_P
73:                     L^GSAR_CENTRAL ← load L^GSAR_CENTRAL generated by GKGA at S_CENTRAL
74:                     add L^GSAR_CENTRAL to Briefcase
75:                     add updated Briefcase to AgentProfile
76:                     for i ← 1, L^NODES.size do          ▷ for each node in the itinerary
77:                         AgentVersion ← i
78:                         add AgentVersion to AgentProfile
79:                         NodeAddress ← L^NODES.get(i)          ▷ get an IP address
80:                         AObject[i] ← new GKDA_P(AgentProfile)
81:                         Add AObject[i] to AgentProfile          ▷ clone's state
82:                         Transfer AgentProfile to DM_AEE at NodeAddress
83:                     end for
84:                 end case
85:             end switch
86:         end if
87:     end case
88:     case result                                  ▷ process the results of mobile agent
89:         AgentName ← read mobile agent's name
90:         ItinType ← read mobile agent's ItinType
91:         add AgentName to L^AgentInfo
92:         add ItinType to L^AgentInfo
93:         ▷ Result processing for Serial Itinerary Agents
94:         if ItinType = "Serial" then
95:             AgentProfile ← contact RM for L^AgentInfo
96:             Briefcase ← retrieve Briefcase from AgentProfile
97:             switch AgentName do
98:                 case LFIGA
99:                     process the Briefcase of LFIGA
100:                end case
101:                case LKGA
102:                    process the Briefcase of LKGA
103:                end case
104:                case TFICA
105:                    call GFIGA(Briefcase)          ▷ stationary agent
106:                end case
107:                case LKCA
108:                    call GKGA(Briefcase)          ▷ stationary agent
109:                end case
110:                case GKDA
111:                    process the Briefcase of GKDA
112:                end case
113:            end switch
114:        end if                                          ▷ continue...
```

AL - continued

```
        ▷ Result processing for Parallel Itinerary Agents
        if ItinType = "Parallel" then
            L_AgentName^AllProfile ← contact RM for L^AgentInfo
            switch AgentName do
                case LKGA_P
                    for all AgentProfile ∈ L_AgentName^AllProfile do        ▷ for each clone
                        Briefcase ← retrieve Briefcase from AgentProfile
                        process the Briefcase of LKGA_P clone
                    end for
                end case
                case LKCA_P
                    call RIGKGA(L_AgentName^AllProfile)                     ▷ stationary agent
                end case
                case GKDA_P
                    for all AgentProfile ∈ L_AgentName^AllProfile do        ▷ for each clone
                        Briefcase ← retrieve Briefcase from AgentProfile
                        process the Briefcase of GKDA_P clone
                    end for
                end case
            end switch
        end if
    end case
  end switch
end procedure
```

Algorithm 10 Result Manager (RM)

```
1:  procedure RM( )
2:      while TRUE do
3:          listen and receive the incoming request
4:          if contacted by a mobile agent for submitting results then
5:              AgentProfile ← receive the incominng AgentProfile from site S_i
6:              ItinType ← retrieve ItinType from AgentProfile
7:              Briefcase ← retrieve mobile agent's Briefcase from AgentProfile
8:              TripTime_start ← retrieve TripTime_start from Briefcase
9:              TripTime_end ← retrieve TripTime_end from Briefcase
10:             TripTime ← TripTime_end − TripTime_start
11:             add TripTime to Briefcase
12:             add updated Briefcase to AgentProfile
13:             if ItinType = "Serial" then
14:                 save AgentProfile at S_CENTRAL
15:             end if                                                  ▷ continue...
```

RM - continued

```
            if ItinType = "Parallel" then
                AgentName ← retrieve AgentName from AgentProfile
                AgentVersion ← retrieve AgentVersion from AgentProfile
                if AgentVersion = 1 then
                    add AgentProfiletoL^AllProfiles_AgentName
                    save L^AllProfiles_AgentName at S_CENTRAL
                end if
                if AgentVersion > 1 then
                    retrieve L^AllProfiles_AgentName from S_CENTRAL
                    add AgentProfile to L^AllProfiles_AgentName
                    save updated L^AllProfiles_AgentName S_CENTRAL
                end if
            end if
        end if
        if contacted by AgentLauncher for processing the results then
            AgentName ← retrieve AgentName from incomming L^AgentInfo
            ItinType ← retrieve ItinType from incomming L^AgentInfo
            if ItinType = "Serial" then
                AgentProfile ← load AgentProfile for AgentName from S_CENTRAL
                dispatch AgentProfile to AgentLauncher
            else
                L^AllProfiles_AgentName ← load L^AllProfiles_AgentName from S_CENTRAL
                dispatch L^AllProfiles_AgentName to AgentLauncher
            end if
        end if
    end while
end procedure
```

The overall working of AeMGSAR may be divided into following six stages:

1. ***Request Stage***: Request for DARM is initiated by AL at $S_{CENTRAL}$ on behalf of the user with necessary credentials.

2. ***Preparation Stage***: AL through User Interface reads agent name, version number; Itinerary (serial or parallel) for the MAs journey is obtained in terms of the IP address of the distributed nodes to be visited by a MA; any specific additional data for a specific MA is obtained; Agent code for the specific MA is loaded from Agent Pool; for serial itinerary a single specific MA is dispatched by AL to travel and visit n distributed sites; for parallel itinerary clones of a specific MA is dispatched by AL to visit n distributed sites in parallel.

3. ***Local Mining Stage***: ARM process is performed locally by specific DM agents on each distributed site and results are kept as local knowledge base at that site.

4. ***Result Collection Stage***: Collector agents visit each site and collect the results generated by DM agents and submit the results back to RM at $S_{CENTRAL}$.

5. ***Knowledge Integration and Global Knowledge Generation Stage***: Knowledge or result integration is carried out by the *RM* with the help of stationary agent and Global Knowledge in the form of Globally Strong Association Rules may be generated with the help of other stationary agents at $S_{CENTRAL}$.

6. ***Global Knowledge Dispatching Stage***: Global Knowledge is dispatched to the distributed sites by a dispatcher agent to compare it with the local knowledge at each site.

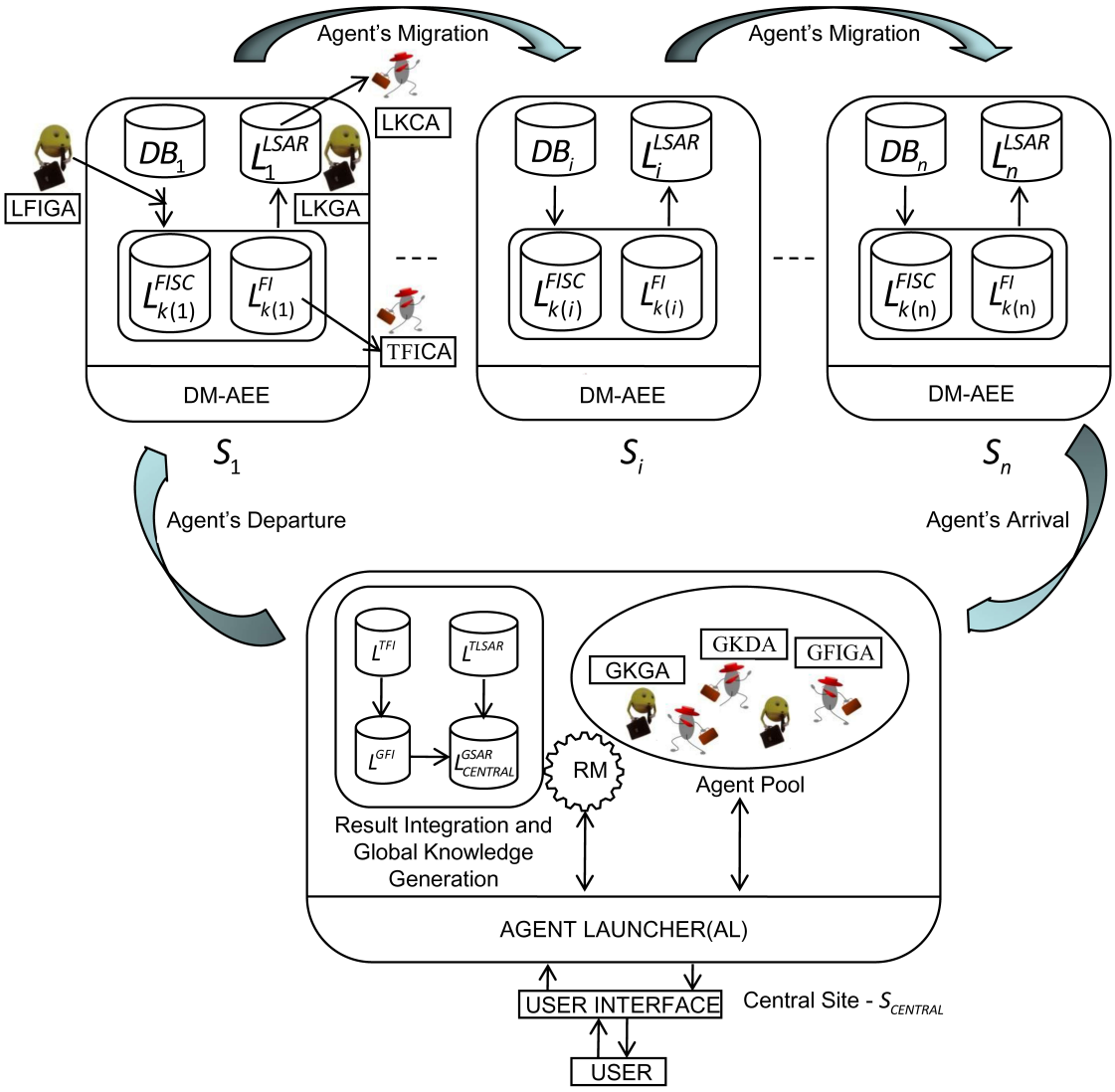

Figure 4.1: AeMGSAR Serial Computing Model

4.2 Serial Computing Model of AeMGSAR

Serial computing model of AeMGSAR system is shown in Fig. 4.1. It consists of total seven agents, five of these are MAs dispatched from $S_{CENTRAL}$ with serial itinerary multi-hop migration and other two are intelligent SAs running at $S_{CENTRAL}$ to perform different

tasks. The CPU time taken by a MA while processing on each site along with some other specific information is carried back in the result bag at $S_{CENTRAL}$. Agents in serial number 1-5 visit n sites serially and other parameters are collected from different resources. Detailed Relationship and working behavior of each agent is as follows.

1. ***Local Frequent Itemset Generater Agent (LFIGA)***: This is a MA that carries the $AgentProfile$ & min_th_sup. $LFIGA$ generates and stores $L^{FI}_{k(i)}$ and $L^{FISC}_{k(i)}$ at site S_i by scanning the local DB_i at that site with the constraint of min_th_sup. It carries back the computational time ($CPUTime$) at each site S_i and $TripTime_{end}$. This agent is embedded with Apriori algorithm [5] for generating all the frequent k-itemset lists. It makes use of a module described in Algorithm 5 in Chapter 3 to generate candidate frequent k-itemset list by joining a frequent (k-1)-itemset with itself. It is may be equipped with decision making capability to select other FIM algorithms based on the density of the dataset at a particular site. More details are available in Algorithm 11.

2. ***Local Knowledge Generater Agent (LKGA)***: This is a MA that carries the $AgentProfile$ & min_th_conf. $LKGA$ applies the constraint of min_th_conf to generate and store L^{LSAR}_i by using the $L^{FI}_{k(i)}$ and $L^{FISC}_{k(i)}$ lists already generated by $LFIGA$ agent at site S_i. L^{LSAR}_i list also contains support and confidence for a particular association rule along with the site name. It carries back the computational time ($CPUTime$) at each site S_i and $TripTime_{end}$. Detailed steps are given in Algorithm 12.

3. ***Total Frequent Itemset Collector Agent (TFICA)***: This is a MA that carries the $AgentProfile$. $TFICA$ collects lists of local frequent k-itemset ($L^{FI}_{k(i)}$) generated by $LFIGA$ agent and carries back the list of total frequent k-itemset (L^{TFI}_k) in the result bag to RM at $S_{CENTRAL}$. In addition to this resultant knowledge it also carries back the computational time ($CPUTime$) at each site S_i and $TripTime_{end}$ and executes Algorithm 13.

4. ***Local Knowledge Collector Agent (LKCA)***: This is a MA that carries the $AgentProfile$. $LKCA$ collects the list of locally strong association rules (L^{LSAR}_i) generated by $LKGA$ agent and carries back the list of total locally strong association rules(L^{TLSAR}) in the result bag to RM at $S_{CENTRAL}$. In addition to this resultant knowledge it also carries back the computational time ($CPUTime$) at each site S_i and $TripTime_{end}$. Steps are shown in Algorithm 14.

5. ***Global Knowledge Dispatcher Agent (GKDA)***: This is a MA that carries the $AgentProfile$ containing global knowledge ($L^{GSAR}_{CENTRAL}$). It dispatches global knowledge at every site for further decision making and comparing with the local knowledge at that site. It executes Algorithm 17.

6. ***Global Frequent Itemset Generater Agent (GFIGA)***: It is a stationary agent at $S_{CENTRAL}$, mainly used for processing the result bag of $TFICA$, i.e., total frequent k-itemset list (L^{TFI}_k) generated by $TFICA$ to generate the global frequent itemset list (L^{GFI}_k). More details are available in Algorithm 15.

7. ***Global Knowledge Generater Agent (GKGA)***: It is also a stationary agent at $S_{CENTRAL}$, mainly used for processing the L_k^{GFI} list and L^{TLSAR} list to compile the global knowledge, i.e., the list of globally strong association rules ($L_{CENTRAL}^{GSAR}$). Detailed steps are shown in Algorithm 16.

Algorithm 11 Local Frequent Itemset Generater Agent (LFIGA)

Input:

- $AgentProfile$, *A collection of agent attributes set by the AL*
- min_th_sup, *the given minimum threshold support*

Output: $L^{FI\&SC}$, *the list of frequent itemsets and their support counts*

1: **procedure** LFIGA($AgentProfile, min\_th\_sup$)
2: $CPUTime_{start} \leftarrow$ *get system time*
3: $Briefcase \leftarrow$ *get Briefcase from AgentProfile*
4: $DB_i \leftarrow$ *load* DB_i *from local file system of site* S_i
5: $T \leftarrow DB_i.get(0)$ ▷ No. of records
6: $I \leftarrow DB_i.get(1)$ ▷ No. of items
7: $DB[T][I] \leftarrow DB_i.get(3)$ ▷ itemset data bank
8: $minsupcount \leftarrow (T \times min\_th\_sup)/100$
9: ▷ generate frequent-1 itemset list (FIL_1) and support count list ($FISC_1$)
10: $CFIL_1 \leftarrow \{1, 2, 3 \cdots I\}$ ▷ candidate frequent-1 itemset
11: **for** $i \leftarrow 1, I$ **do** ▷ initialize the support count array $SCFIL_1$ to zero
12: $SCFIL_1[i] \leftarrow 0$
13: **end for**
14: $k \leftarrow 1$
15: **for all** *candidate* $c \in CFIL_1$ **do** ▷ find support count for every candidate
16: **for all** *transaction* $t \in DB$ **do** ▷ scan DB
17: **if** $c \subset t$ **then**
18: $SCFIL_1[k] \leftarrow SCFIL_1[k] + 1$
19: **end if**
20: **end for**
21: $k \leftarrow k + 1$
22: **end for**
23: ▷ prune $CFIL_1$ to generate FIL_1 and $FISC_1$
24: **for** $k \leftarrow 1, I$ **do**
25: **if** $SCFIL_1[k] \geq minsupcount$ **then**
26: *add* $c_k \in CFIL_1$ *to* FIL_1
27: *add* $SCFIL_1[k]$ *to* $FISC_1$
28: **end if**
29: **end for**
30: **if** $FIL_1 \neq \emptyset$ **then**
31: *add* FIL_1 *to* L^{FI}
32: *add* $FISC_1$ *to* L^{FISC}
33: **end if** ▷ continue...

LFIGA - continued

```
34:  k ← 2
35:  while FIL_{k-1} ≠ ∅ do
36:      CFIL_k ← Call GenerateCFIL(FIL_{k-1})   ▷ see Algorithm 5 in chapter 3
37:      for i ← 1, CFIL_k.length do   ▷ initialize the array SCFIL_k to zero
38:          SCFIL_k[i] ← 0
39:      end for
40:      i ← 1
41:      for all candidate c ∈ CFIL_k do   ▷ find support count for every candidate
42:          for all transaction t ∈ DB do   ▷ scan DB
43:              if c ⊂ t then
44:                  SCFIL_k[i] ← SCFIL_k[i] + 1
45:              end if
46:          end for
47:          i ← i + 1
48:      end for
49:      ▷ prune CFIL_k to generate FIL_k and FISC_k
50:      for i ← 1, SCFIL_k.length do
51:          if SCFIL_k[i] ≥ minsupcount then
52:              add c_i ∈ CFIL_k to FIL_k
53:              add SCFIL_k[i] to FISC_k
54:          end if
55:      end for
56:      if FIL_k ≠ ∅ then
57:          add FIL_k to L^FI
58:          add FISC_k to L^FISC
59:      end if
60:      k ← k + 1
61:  end while
62:  add T to L^FI&SC
63:  add L^FI to L^FI&SC
64:  add L^FISC to L^FI&SC
65:  save L^FI&SC in the local file system of this site S_i
66:  CPUTime_end ← get system time
67:  CPUTime ← CPUTime_end − CPUTime_start
68:  add CPUTime to Result_S_i
69:  add Result_S_i to Briefcase
70:  add updated Briefcase to AgentProfile
71:  L^NODES ← get itinerary list from AgentProfile
72:  L^NODES ← remove first IP address from L^NODES   ▷ visited site
73:  add updated L^NODES to AgentProfile
74:  if L^NODES ≠ ∅ then   ▷ itinerary not empty
75:      AObject ← new LFIGA(AgentProfile, min_th_sup)
76:      add AObject to AgentProfile
77:      transfer AgentProfile to DM_AEE at first IP address in L^NODES
78:  else   ▷ continue...
```

LFIGA - continued

$TripTime_{end} \leftarrow$ *get system time for end of agent journey*
add $TripTime_{end}$ *to* $Briefcase$
add updated $Briefcase$ *to* $AgentProfile$
transfer $AgentProfile$ *to RM at* $S_{CENTRAL}$
end if
end procedure

Algorithm 12 Local Knowledge Generater Agent (LKGA)

Input:

- $AgentProfile$, *A collection of Agent attributes set by the AL*
- min_th_conf, *the given minimum threshold confidence*

Output: L^{LSAR}, *the list of locally strong association rules*

procedure LKGA($AgentProfile, min\_th\_conf$)
$CPUTime_{start} \leftarrow$ *get system time*
$Briefcase \leftarrow$ *get* $Briefcase$ *from* $AgentProfile$
$L^{FI\&SC} \leftarrow$ *load* $L^{FI\&SC}$ *from local file system of this site* S_i
$T \leftarrow L^{FI\&SC}.get(0)$ ▷ No. of records
$L^{FI} \leftarrow L^{FI\&SC}.get(1)$ ▷ frequent k-itemset list
$L^{FISC} \leftarrow L^{FI\&SC}.get(2)$ ▷ support count list
for $k \leftarrow 2, L^{FI}.size$ **do**
$L_k \leftarrow L^{FI}.get(k)$ ▷ get frequent k-itemset list
for all $l \in L_k$ **do**
$l_{subsets} \leftarrow$ *generate all non − empty subsets of* l
$l_{spcount} \leftarrow$ *get support count of* l *from* L^{FISC}
$AR_{support} \leftarrow (l_{spcount}/T) \times 100$ ▷ support of the association rule
for all *non − empty subset* $s \in l_{subsets}$ **do**
$s_{spcount} \leftarrow$ *get support count of* s *from* L^{FISC}
$AR_{conf} \leftarrow (l_{spcount}/s_{spcount}) \times 100$ ▷ confidence of the association rule
if $AR_{conf} \geq min\_th\_conf$ **then**
$AR_{strong} \leftarrow$ "$s \Rightarrow l - s[AR_{support}\%, AR_{conf}\%]$"
print AR_{strong}
add l *to* AR_{strong}
$S_i^{IP} \leftarrow$ *get IP address of this site* S_i
add S_i^{IP} *to* AR_{strong}
add AR_{strong} *to* L^{LSAR}
end if
end for
end for
end for
save L^{LSAR} *in the local file system of this site* S_i
$CPUTime_{end} \leftarrow$ *get system time*
$CPUTime \leftarrow CPUTime_{end} - CPUTime_{start}$ ▷ continue...

LKGA - continued

31: *add CPUTime to Result_S_i and Result_S_i to Briefcase*
32: *add updated Briefcase to AgentProfile*
33: $L^{NODES} \leftarrow$ *get itinerary list from AgentProfile*
34: $L^{NODES} \leftarrow$ *remove first IP address from* L^{NODES} ▷ visited site
35: *add updated* L^{NODES} *to AgentProfile*
36: **if** $L^{NODES} \neq \emptyset$ **then** ▷ itinerary not empty
37: *AObject* $\leftarrow$ *new LKGA(AgentProfile, min_th_conf)*
38: *add AObject to AgentProfile*
39: *transfer AgentProfile to DM_AEE at first IP address in* L^{NODES}
40: **else**
41: $TripTime_{end} \leftarrow$ *get system time for end of agent journey*
42: *add* $TripTime_{end}$ *to Briefcase and updated Briefcase to AgentProfile*
43: *transfer AgentProfile to RM at* $S_{CENTRAL}$
44: **end if**
45: **end procedure**

Algorithm 13 Total Frequent Itemset Collector Agent (TFICA)

Input: *AgentProfile, A collection of Agent attributes set by the AL*
Output: L^{FI}, *the list of locally frequent itemsets*

1: **procedure** TFICA(*AgentProfile*)
2: $CPUTime_{start} \leftarrow$ *get system time*
3: *Briefcase* $\leftarrow$ *get Briefcase from AgentProfile*
4: $L^{FI\&SC} \leftarrow$ *load* $L^{FI\&SC}$ *from local file system of this site* S_i
5: $L^{FI} \leftarrow L^{FI\&SC}.get(1)$ ▷ frequent k-itemset list
6: *add* L^{FI} *to Result_S_i*
7: $CPUTime_{end} \leftarrow$ *get system time*
8: $CPUTime \leftarrow CPUTime_{end} - CPUTime_{start}$
9: *add CPUTime to Result_S_i and Result_S_i to Briefcase*
10: *add updated Briefcase to AgentProfile*
11: $L^{NODES} \leftarrow$ *get itinerary list from AgentProfile*
12: $L^{NODES} \leftarrow$ *remove first IP address from* L^{NODES} ▷ visited site
13: *add updated* L^{NODES} *to AgentProfile*
14: **if** $L^{NODES} \neq \emptyset$ **then** ▷ itinerary not empty
15: *AObject* $\leftarrow$ *new TFICA(AgentProfile)*
16: *add AObject to AgentProfile*
17: *transfer AgentProfile to DM_AEE at first IP address in* L^{NODES}
18: **else**
19: $TripTime_{end} \leftarrow$ *get system time for end of agent journey*
20: *add* $TripTime_{end}$ *to Briefcase*
21: *add updated Briefcase to AgentProfile*
22: *transfer AgentProfile to RM at* $S_{CENTRAL}$
23: **end if**
24: **end procedure**

Algorithm 14 Local Knowledge Collector Agent (LKCA)

Input: *AgentProfile, A collection of Agent attributes set by the AL*
Output: L^{LSAR}, *the list of locally strong association rules*

1: **procedure** LKCA(*AgentProfile*)
2: $CPUTime_{start} \leftarrow$ *get system time*
3: $Briefcase \leftarrow$ *get Briefcase from AgentProfile*
4: $L^{LSAR} \leftarrow$ *load* L^{LSAR} *from local file system of this site* S_i
5: *add* L^{LSAR} *to* $Result\_S_i$
6: $CPUTime_{end} \leftarrow$ *get system time*
7: $CPUTime \leftarrow CPUTime_{end} - CPUTime_{start}$
8: *add CPUTime to* $Result\_S_i$
9: *add* $Result\_S_i$ *to Briefcase*
10: *add updated Briefcase to AgentProfile*
11: $L^{NODES} \leftarrow$ *get itinerary list from AgentProfile*
12: $L^{NODES} \leftarrow$ *remove first IP address from* L^{NODES} ▷ visited site
13: *add updated* L^{NODES} *to AgentProfile*
14: **if** $L^{NODES} \neq \emptyset$ **then** ▷ itinerary not empty
15: *AObject* $\leftarrow$ *new LKCA(AgentProfile)*
16: *add AObject to AgentProfile*
17: *transfer AgentProfile to DM_AEE at first IP address in* L^{NODES}
18: **else**
19: $TripTime_{end} \leftarrow$ *get system time for end of agent journey*
20: *add* $TripTime_{end}$ *to Briefcase*
21: *add updated Briefcase to AgentProfile*
22: *transfer AgentProfile to RM at* $S_{CENTRAL}$
23: **end if**
24: **end procedure**

Algorithm 15 Global Frequent Itemset Generater Agent (GFIGA)

Input: *Briefcase, Result bag of TFICA agent*
Output: L^{GFI}, *the list of global frequent itemsets*

1: **procedure** GFIGA(*Briefcase*)
2: $CPUTime_{start} \leftarrow$ *get system time*
3: $L^{TFI} \leftarrow$ *retrieve total frequent itemsets*$(\bigcup_{i=1}^{n} L_i^{FI})$ *from Briefcase*
4: $L^{GFI} \leftarrow$ *find global frequent itemsets*$(\bigcap_{i=1}^{n} L_i^{FI})$ *from Briefcase*
5: **print** L^{GFI}
6: *save* L^{GFI} *in the local file system of site* $S_{CENTRAL}$
7: $CPUTime_{end} \leftarrow$ *get system time*
8: $CPUTime \leftarrow CPUTime_{end} - CPUTime_{start}$
9: **print** $CPUTime$
10: **return** L^{GFI}
11: **end procedure**

Algorithm 16 Global Knowledge Generater Agent (GKGA)

Input: $Briefcase$, $Result\ bag\ of\ LKCA\ agent$
Output: $L^{GSAR}_{CENTRAL}$, $the\ list\ of\ globally\ strong\ association\ rules$

1: **procedure** GKGA($Briefcase$)
2: $CPUTime_{start} \leftarrow get\ system\ time$
3: $L^{TLSAR} \leftarrow retrieve\ total\ strong\ rules(\bigcup_{i=1}^{n} L_i^{LSAR})\ from\ Briefcase$
4: $L^{GFI} \leftarrow load\ global\ frequent\ itemsets(L^{GFI})\ from\ S_{CENTRAL}$
5: **for all** $AR_{strong} \in L^{TLSAR}$ **do**
6: $L \leftarrow get\ frequent\ itemset\ from\ AR_{strong}$
7: **if** $L \in L^{GFI}$ **then**
8: **print** $AR_{strong}\ along\ with\ \ (S_i^{IP})$ *and add* AR_{strong} *to* $L^{GSAR}_{CENTRAL}$
9: **end if**
10: **end for**
11: $save\ L^{GSAR}_{CENTRAL}\ in\ the\ local\ file\ system\ of\ site\ S_{CENTRAL}$
12: $CPUTime_{end} \leftarrow get\ system\ time$
13: $CPUTime \leftarrow CPUTime_{end} - CPUTime_{start}$
14: **print** $CPUTime$
15: **return** $L^{GSAR}_{CENTRAL}$
16: **end procedure**

Algorithm 17 Global Knowledge Dispatcher Agent (GKDA)

Input: $AgentProfile$, $A\ collection\ of\ Agent\ attributes\ set\ by\ the\ AL$
Output: $Dispatch\ L^{GSAR}_{CENTRAL}\ at\ each\ distributed\ site\ S_i$

1: **procedure** GKDA($AgentProfile$)
2: $CPUTime_{start} \leftarrow get\ system\ time$
3: $Briefcase \leftarrow get\ Briefcase\ from\ AgentProfile$
4: $L^{GSAR}_{CENTRAL} \leftarrow get\ L^{GSAR}_{CENTRAL}\ from\ Briefcase\ and\ save\ it\ at\ site\ S_i$
5: $CPUTime_{end} \leftarrow get\ system\ time$
6: $CPUTime \leftarrow CPUTime_{end} - CPUTime_{start}$
7: $add\ CPUTime\ to\ Result\_S_i\ and\ Result\_S_i\ to\ Briefcase$
8: $add\ updated\ Briefcase\ to\ AgentProfile$
9: $L^{NODES} \leftarrow get\ itinerary\ list\ from\ AgentProfile$
10: $L^{NODES} \leftarrow remove\ first\ IP\ address\ from\ L^{NODES}$ ▷ visited site
11: $add\ updated\ L^{NODES}\ to\ AgentProfile$
12: **if** $L^{NODES} \neq \emptyset$ **then** ▷ itinerary not empty
13: $AObject \leftarrow new\ GKDA(AgentProfile)\ and\ add\ AObject\ to\ AgentProfile$
14: $transfer\ AgentProfile\ to\ DM\_AEE\ at\ first\ IP\ address\ in\ L^{NODES}$
15: **else**
16: $TripTime_{end} \leftarrow get\ system\ time\ for\ end\ of\ agent\ journey$
17: $add\ TripTime_{end}\ to\ Briefcase\ and\ updated\ Briefcase\ to\ AgentProfile$
18: $transfer\ AgentProfile\ to\ RM\ at\ S_{CENTRAL}$
19: **end if**
20: **end procedure**

4.2.1 Overall Time Model

The overall response time T for the DARM task performed using serial computing model of AeMGSAR system is calculated using the following equation:

$$T = t_{darm} + t_{dki} + t_{gkg} + t_{gkd} + t_{wait} \tag{4.1}$$

where,

- t_{darm} = total time taken for DARM task and is calculated using three estimates, of which first and third are communication costs and second one is computational cost:
  - Agent cost, which is the estimate of the round trip time required for all MAs to travel from $S_{CENTRAL}$ to S_1, then moving from node to node and finally coming at $S_{CENTRAL}$ from last visited site, S_n.
  - Local ARM cost, which is the estimate of the time needed for performing ARM task locally by *LFIGA* and *LKGA* agents at each site S_i.
  - Local Knowledge Transfer cost, which is the estimate for the round trip time needed for *TFICA* and *LKCA* to travel from node to node to collect the results and submitting these results at $S_{CENTRAL}$.
- t_{dki} = time taken to perform distributed knowledge integration by *GFIGA* at $S_{CENTRAL}$.
- t_{gkg} = time taken to perform global knowledge generation by *GKGA* at $S_{CENTRAL}$.
- t_{gkd} = round trip time taken by *GKDA* for dispatching the global knowledge to distributed sites for local analysis.
- t_{wait} = total time period for which system will wait for an event to happen. It varies and depends on the availability of the active node.

This model is not fault tolerant, so a site failure and recovery time is not taken into account.

4.2.1.1 Cost Model involving one Site

Let us now consider the case of i^{th} distributed site S_i where DM has to be performed. Let time taken by a MA to travel from node x to node y is refered by $t_{MobileAgent}(x, y)$. The t_{darm} component in equation 4.1 involving i^{th} distributed site S_i is calculated according to equation 4.2.

$$\begin{aligned} t_{darm} = {} & t_{LFIGA}(S_{CENTRAL}, S_i) + t_{fim}(S_i) + t_{LFIGA}(S_i, S_{CENTRAL}) + \\ & t_{LKGA}(S_{CENTRAL}, S_i) + t_{arm}(S_i) + t_{LKGA}(S_i, S_{CENTRAL}) + \\ & t_{TFICA}(S_{CENTRAL}, S_i) + t_{TFICA}(S_i, S_{CENTRAL}) + \\ & t_{LKCA}(S_{CENTRAL}, S_i) + t_{LKCA}(S_i, S_{CENTRAL}) \end{aligned} \tag{4.2}$$

where,

- $t_{LFIGA}(S_{CENTRAL}, S_i)$ = time taken by *LFIGA* (embedded with mining algorithm and other parameters like min_th_sup) to move from $S_{CENTRAL}$ to S_i.

- $t_{fim}(S_i)$ = time taken for performing FIM task locally by *LFIGA* at S_i and it depends upon the following factors: the size and density of dataset ($Size\_Density\,(DB_i)$), no. of other processes running at the local site ($\#\_of\_other\_Processes\,(S_i)$) and the processors speed at S_i ($Processor\_Speed\,(S_i)$). Therefore,

$$t_{fim}(S_i) \propto \frac{k \times Size\_Density(DB_i) \times \#\_of\_other\_Processes(S_i)}{Processor\_Speed(S_i)} \tag{4.3}$$

  where, k is a constant.

- $t_{LFIGA}(S_i, S_{CENTRAL})$ = time taken by *LFIGA* to move from S_i back to $S_{CENTRAL}$.

- $t_{LKGA}(S_{CENTRAL}, S_i)$ = time taken by *LKGA* (embedded with mining algorithm and other parameters like min_th_conf) to move from $S_{CENTRAL}$ to S_i.

- $t_{arm}(S_i)$ = time taken for performing ARM task locally by *LKGA* at S_i and it depends upon the following factors: size of $L^{FI}_{k(i)}$ and $L^{FISC}_{k(i)}$ lists generated by *LFIGA*, i.e., $Size\left(L^{FI}_{k(i)}\right)$ and $Size\left(L^{FISC}_{k(i)}\right)$, no. of other processes running at the local site ($\#\_of\_other\_Processes\,(S_i)$) and the processors speed at S_i ($Processor\_Speed\,(S_i)$). Therefore,

$$t_{arm}\,(S_i) \propto \frac{k \times Size\left(L^{FI}_{k(i)}\right) \times Size\left(L^{FISC}_{k(i)}\right) \times \#\_of\_other\_Processes\,(S_i)}{Processor\_Speed\,(S_i)} \tag{4.4}$$

  where, k is a constant.

- $t_{LKGA}(S_i, S_{CENTRAL})$ = time taken by *LKGA* to move from S_i back to $S_{CENTRAL}$.

- $t_{TFICA}(S_{CENTRAL}, S_i)$ = time taken by *TFICA* to move from $S_{CENTRAL}$ to S_i to collect the desired result.

- $t_{TFICA}(S_i, S_{CENTRAL})$ = time taken by *TFICA* to move from S_i back to $S_{CENTRAL}$ and submit the desired results to *RM*.

- $t_{LKCA}(S_{CENTRAL}, S_i)$ = time taken by *LKCA* to move from $S_{CENTRAL}$ to S_i to collect the desired result.

- $t_{LKCA}(S_i, S_{CENTRAL})$ = time taken by *LKCA* to move from S_i back to $S_{CENTRAL}$ and submit the desired results to *RM*.

In this case $t_{dki} = 0$, as there is no need for knowledge integration for this particular case and the result bag of *LKCA* can be directly processed by *GKGA* for generating global knowledge. The t_{gkd} component in equation 4.1 is calculated using equation 4.5.

$$t_{gkd} = t_{GKDA}(S_{CENTRAL}, S_i) + t_{GKDA}(S_i, S_{CENTRAL}) \tag{4.5}$$

Putting equation 4.2 and 4.5 in equation 4.1, total response time T for DARM task for this case can be expressed as:

$$\begin{aligned} T = t_{LFIGA}(S_{CENTRAL}, S_i) + t_{fim}(S_i) + t_{LFIGA}(S_i, S_{CENTRAL}) + \\ t_{LKGA}(S_{CENTRAL}, S_i) + t_{arm}(S_i) + t_{LKGA}(S_i, S_{CENTRAL}) + \\ t_{TFICA}(S_{CENTRAL}, S_i) + t_{TFICA}(S_i, S_{CENTRAL}) + \\ t_{LKCA}(S_{CENTRAL}, S_i) + t_{LKCA}(S_i, S_{CENTRAL}) + t_{gkg} + \\ t_{GKDA}(S_{CENTRAL}, S_i) + t_{GKDA}(S_i, S_{CENTRAL}) + t_{wait} \end{aligned} \tag{4.6}$$

In general, $t_{MobileAgent}(x, y)$ depends upon the size of the agent ($Size_{MobileAgent}$) and the network bandwidth (in Kbits per second) between two nodes x and y ($BWidth(x, y)$). Therefore,

$$t_{LFIGA}(S_{CENTRAL}, S_i) \propto \frac{k \times Size_{LFIGA}}{BWidth(S_{CENTRAL}, S_i)} \tag{4.7}$$

where k is a constant.

Size of agent is defined by the triplet,
$Size_{MobileAgent} = \langle Agent\ State, Agent\ Code, Agent\ Data \rangle$[106] where, Agent State is the execution state of MA, Agent Code is the program encapsulated within an agent to perform its functionality and Agent Data includes either the intermediate or final results of some computation performed at a remote site or the additional input parameters required for Agent Code. These considerations defines the following equations:

$$Size_{LFIGA} = \langle LFIGA\ State, FIMAlgorithm, InputParameters \rangle \tag{4.8}$$

$$Size_{LKGA} = \langle LKGA\ State, ARMAlgorithm, InputParameters \rangle \tag{4.9}$$

$t_{LFIGA}(S_i, S_{CENTRAL})$, $t_{LKGA}(S_{CENTRAL}, S_i)$, $t_{LKGA}(S_i, S_{CENTRAL})$, $t_{TFICA}(S_{CENTRAL}, S_i)$, $t_{TFICA}(S_i, S_{CENTRAL})$, $t_{LKCA}(S_{CENTRAL}, S_i)$, $t_{LKCA}(S_i, S_{CENTRAL})$, $t_{GKDA}(S_{CENTRAL}, S_i)$,and $t_{GKDA}(S_i, S_{CENTRAL})$ is also calculated similar to $t_{LFIGA}(S_{CENTRAL}, S_i)$ except that $Size_{TFICA}$ and $Size_{LKCA}$ does not include any DARM algorithm code and Local knowledge is included as Agent Data component and in $Size_{GKDA}$ includes the Global knowledge as Agent Data Component.

4.2.1.2 Cost Model involving n Distributed Sites

AL dispatches single copy of each MA in sequence to n distributed sites. Mining is performed locally at each distributed site in sequence from node to node by *LFIGA* and *LKGA*, then results are transferred through *TFICA* and *LKCA* to $S_{CENTRAL}$. Since mining is performed locally at each site sequentially, the total time taken depends upon the time interval required by the site that takes longest time to perform DM and return

results. Therefore,

$$t_{darm} = t_{LFIGA}(S_{CENTRAL}, S_1) + \sum_{i=1}^{n-1} t_{LFIGA}(S_i, S_{i+1}) + \max(t_{fim}(S_i)) + t_{LFIGA}(S_n, S_{CENTRAL}) + t_{LKGA}(S_{CENTRAL}, S_1) + \sum_{i=1}^{n-1} t_{LKGA}(S_i, S_{i+1}) + \max(t_{arm}(S_i)) + t_{LKGA}(S_n, S_{CENTRAL}) + t_{TFICA}(S_{CENTRAL}, S_1) + \sum_{i=1}^{n-1} t_{TFICA}(S_i, S_{i+1}) + t_{TFICA}(S_n, S_{CENTRAL}) + t_{LKCA}(S_{CENTRAL}, S_1) + \sum_{i=1}^{n-1} t_{LKCA}(S_i, S_{i+1}) + t_{LKCA}(S_n, S_{CENTRAL}) \quad (4.10)$$

The t_{gkd} component in equation 4.1 is calculated using equation 4.11.

$$t_{gkd} = t_{GKDA}(S_{CENTRAL}, S_1) + \sum_{i=1}^{n-1} t_{GKDA}(S_i, S_{i+1}) + t_{GKDA}(S_n, S_{CENTRAL}) \quad (4.11)$$

Putting equation 4.10 and 4.11 in equation 4.1, overall response time T for the DARM task is given as:

$$T = t_{LFIGA}(S_{CENTRAL}, S_1) + \sum_{i=1}^{n-1} t_{LFIGA}(S_i, S_{i+1}) + \max(t_{fim}(S_i)) + t_{LFIGA}(S_n, S_{CENTRAL}) + t_{LKGA}(S_{CENTRAL}, S_1) + \sum_{i=1}^{n-1} t_{LKGA}(S_i, S_{i+1}) + \max(t_{arm}(S_i)) + t_{LKGA}(S_n, S_{CENTRAL}) + t_{TFICA}(S_{CENTRAL}, S_1) + \sum_{i=1}^{n-1} t_{TFICA}(S_i, S_{i+1}) + t_{TFICA}(S_n, S_{CENTRAL}) + t_{LKCA}(S_{CENTRAL}, S_1) + \sum_{i=1}^{n-1} t_{LKCA}(S_i, S_{i+1}) + t_{LKCA}(S_n, S_{CENTRAL}) + t_{dki} + t_{gkg} + t_{GKDA}(S_{CENTRAL}, S_1) + \sum_{i=1}^{n-1} t_{GKDA}(S_i, S_{i+1}) + t_{GKDA}(S_n, S_{CENTRAL}) + t_{wait}, \forall i = \{1 \dots n\} \quad (4.12)$$

4.2.2 Implementation and Performance Study

All the agents are designed in Java language. The datasets are same as defined in subsection 3.3.2 in Chapter 3. The required configuration for the framework is same as defined in Table 3.2 in Chapter 3 with additional deployment of *DM_AEE* at each distributed site and *AL* and *RM* at $S_{CENTRAL}$. Round Trip time taken by various MAs is shown

in Fig. 4.2. CPU time consumed by various MAs at site S_1, S_2 and S_3 is shown in Fig. 4.3, Fig. 4.4 and Fig. 4.5, respectively. CPU time for *GFIGA* and *GKGA* is 101357102 nano seconds and 33317458 nano seconds, respectively. $L^{FI}_{k(i)}$ and $L^{FISC}_{k(i)}$ at distributed sites generated by *LFIGA* agent with 20% *min_th_sup* are shown in Appendix B.1, B.2 and B.3. L^{LSAR}_i at distributed sites generated by *LKGA* agent with 50% *min_th_conf* are shown in Appendix B.4, B.5 and B.6. Globally frequent itemsets generated by *GFIGA* at $S_{CENTRAL}$ is shown in Fig. 4.6. Fifteen numbers of 2-itemsets and eight numbers of 3-itemsets are globally frequent in L^{TFI}_k list and 4, 5, and 6-itemsets, which are locally frequent, are not globally frequent. Globally strong association rules ($L^{GSAR}_{CENTRAL}$) generated by *GKGA* at $S_{CENTRAL}$ for globally frequent 3-itemsets are shown in Fig. 4.7 and $L^{GSAR}_{CENTRAL}$ for 2-itemsets are shown in Appendix B.7.

Table 4.1 highlights the comparative analysis of the serial computing model of the proposed framework AeMGSAR with the central DW based ARM based on the same parameters as in Table 3.3 in Chapter 3. Serial itinerary used for MAs increases the overall cost of DARM task so a parallel computing model of AeMGSAR is designed.

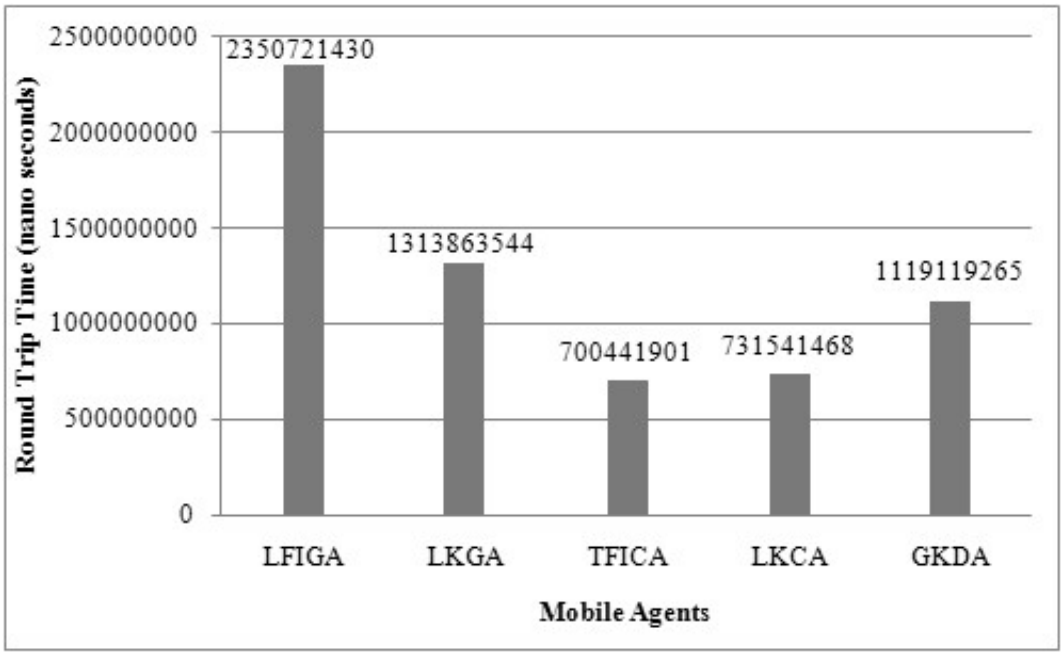

Figure 4.2: Round Trip Time taken by various MAs

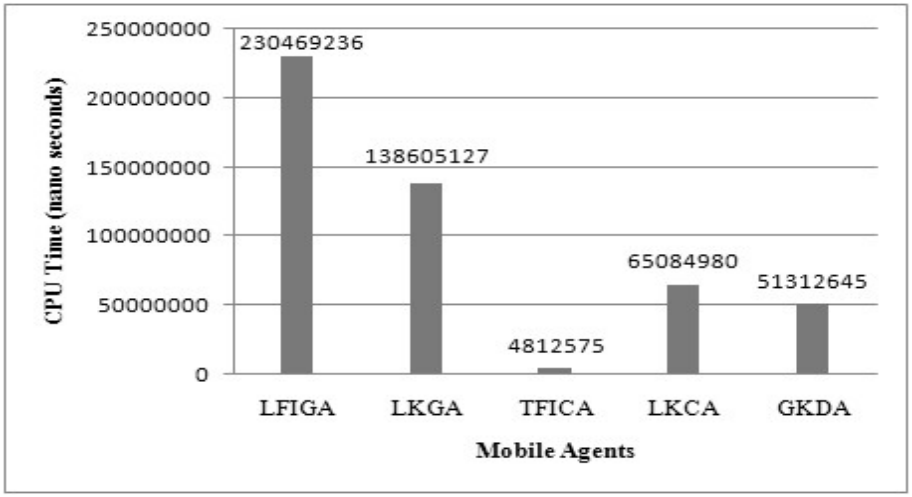

Figure 4.3: CPU Time taken by various MAs at site S_1

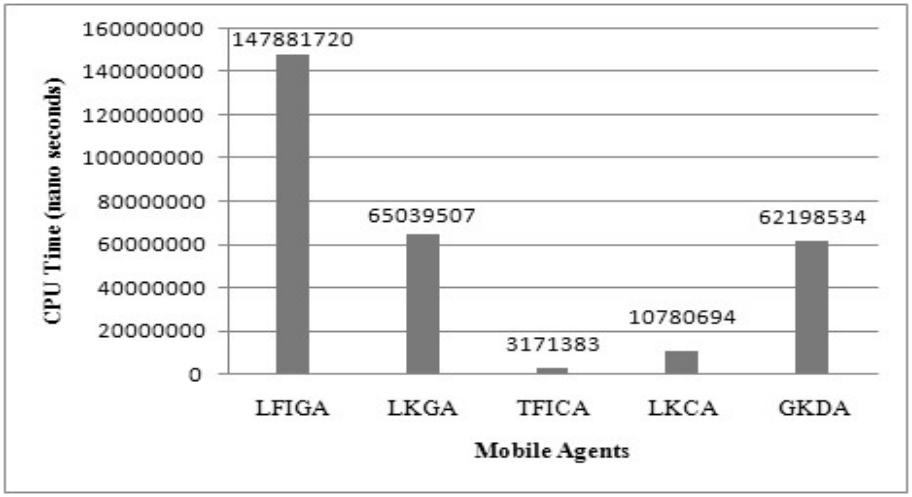

Figure 4.4: CPU Time taken by various MAs at site S_2

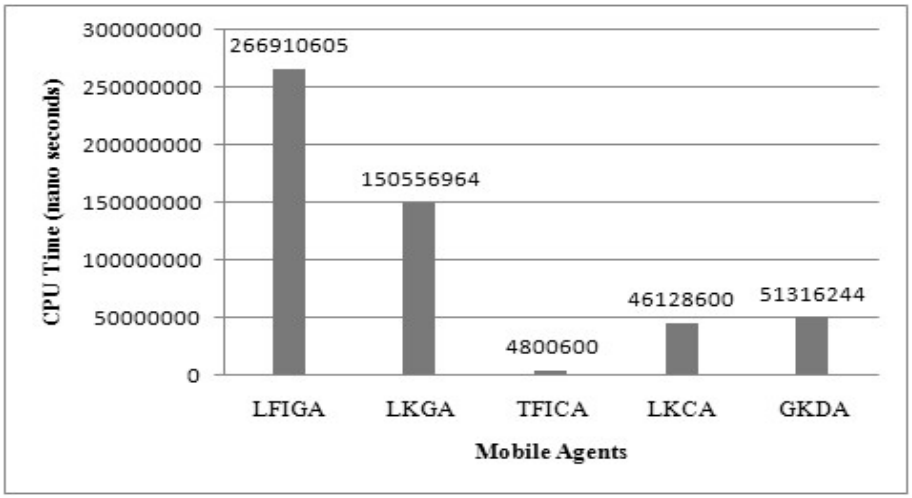

Figure 4.5: CPU Time taken by various MAs at site S_3

Global Frequent Itemsets		
S.n.	2-Itemsets	3-Itemsets
	L	L
1	[1,3]	[1,4,7]
2	[1,4]	[1,4,8]
3	[1,7]	[1,4,9]
4	[1,8]	[1,7,8]
5	[1,9]	[1,8,9]
6	[3,4]	[4,7,9]
7	[3,7]	[4,8,9]
8	[3,9]	[7,8,9]
9	[4,7]	
10	[4,8]	
11	[4,9]	
12	[6,8]	
13	[7,8]	
14	[7,9]	
15	[8,9]	

Figure 4.6: Lists of global frequent k-itemsets at $S_{CENTRAL}$

Stong Association Rules for 3-Itemsets			
S.n.	L	AR (support,confidence)	Site
1	[1, 4, 7]	[4] => [1, 7] (44%, 70%)	192.168.46.212
		[7] => [1, 4] (44%, 62%)	192.168.46.212
		[1, 4] => [7] (44%, 71%)	192.168.46.212
		[1, 7] => [4] (44%, 63%)	192.168.46.212
		[4, 7] => [1] (44%, 98%)	192.168.46.212
		[4] => [1, 7] (27%, 51%)	192.168.46.189
		[1, 4] => [7] (27%, 67%)	192.168.46.189
		[1, 7] => [4] (27%, 53%)	192.168.46.189
		[4, 7] => [1] (27%, 76%)	192.168.46.189
		[1, 7] => [4] (20%, 76%)	192.168.46.213
		[4, 7] => [1] (20%, 53%)	192.168.46.213
2	[1, 4, 8]	[8] => [1, 4] (21%, 63%)	192.168.46.212
		[1, 8] => [4] (21%, 64%)	192.168.46.212
		[4, 8] => [1] (21%, 98%)	192.168.46.212
		[1] => [4, 8] (39%, 51%)	192.168.46.189
		[4] => [1, 8] (39%, 74%)	192.168.46.189
		[1, 4] => [8] (39%, 97%)	192.168.46.189
		[1, 8] => [4] (39%, 53%)	192.168.46.189
		[4, 8] => [1] (39%, 76%)	192.168.46.189
		[1] => [4, 8] (40%, 75%)	192.168.46.213
		[4] => [1, 8] (40%, 52%)	192.168.46.213
		[1, 4] => [8] (40%, 98%)	192.168.46.213
		[1, 8] => [4] (40%, 76%)	192.168.46.213
		[4, 8] => [1] (40%, 53%)	192.168.46.213
3	[1, 4, 9]	[1] => [4, 9] (57%, 58%)	192.168.46.212
		[4] => [1, 9] (57%, 90%)	192.168.46.212
		[9] => [1, 4] (57%, 62%)	192.168.46.212
		[1, 4] => [9] (57%, 91%)	192.168.46.212
		[1, 9] => [4] (57%, 63%)	192.168.46.212
		[4, 9] => [1] (57%, 98%)	192.168.46.212
		[1, 4] => [9] (25%, 62%)	192.168.46.189
		[1, 9] => [4] (25%, 53%)	192.168.46.189
		[4, 9] => [1] (25%, 75%)	192.168.46.189
		[1, 4] => [9] (25%, 61%)	192.168.46.213
		[1, 9] => [4] (25%, 76%)	192.168.46.213
		[4, 9] => [1] (25%, 53%)	192.168.46.213

Stong Association Rules for 3-Itemsets			
S.n.	L	AR (support,confidence)	Site
4	[1, 7, 8]	[8] => [1, 7] (24%, 70%)	192.168.46.212
		[1, 8] => [7] (24%, 71%)	192.168.46.212
		[7, 8] => [1] (24%, 98%)	192.168.46.212
		[1] => [7, 8] (49%, 64%)	192.168.46.189
		[7] => [1, 8] (49%, 74%)	192.168.46.189
		[8] => [1, 7] (49%, 51%)	192.168.46.189
		[1, 7] => [8] (49%, 96%)	192.168.46.189
		[1, 8] => [7] (49%, 66%)	192.168.46.189
		[7, 8] => [1] (49%, 76%)	192.168.46.189
		[7] => [1, 8] (25%, 52%)	192.168.46.213
		[1, 7] => [8] (25%, 98%)	192.168.46.213
		[7, 8] => [1] (25%, 53%)	192.168.46.213
5	[1, 8, 9]	[8] => [1, 9] (30%, 89%)	192.168.46.212
		[1, 8] => [9] (30%, 91%)	192.168.46.212
		[8, 9] => [1] (30%, 98%)	192.168.46.212
		[1] => [8, 9] (46%, 59%)	192.168.46.189
		[9] => [1, 8] (46%, 73%)	192.168.46.189
		[1, 8] => [9] (46%, 61%)	192.168.46.189
		[1, 9] => [8] (46%, 96%)	192.168.46.189
		[8, 9] => [1] (46%, 76%)	192.168.46.189
		[1] => [8, 9] (32%, 60%)	192.168.46.213
		[9] => [1, 8] (32%, 53%)	192.168.46.213
		[1, 8] => [9] (32%, 61%)	192.168.46.213
		[1, 9] => [8] (32%, 98%)	192.168.46.213
		[8, 9] => [1] (32%, 53%)	192.168.46.213
6	[4, 7, 9]	[4] => [7, 9] (41%, 64%)	192.168.46.212
		[7] => [4, 9] (41%, 58%)	192.168.46.212
		[4, 7] => [9] (41%, 91%)	192.168.46.212
		[4, 9] => [7] (41%, 70%)	192.168.46.212
		[7, 9] => [4] (41%, 64%)	192.168.46.212
		[4, 7] => [9] (23%, 64%)	192.168.46.189
		[4, 9] => [7] (23%, 69%)	192.168.46.189
		[7, 9] => [4] (23%, 54%)	192.168.46.189
		[4, 7] => [9] (22%, 60%)	192.168.46.213
		[7, 9] => [4] (22%, 75%)	192.168.46.213

Stong Association Rules for 3-Itemsets			
S.n.	L	AR (support,confidence)	Site
7	[4, 8, 9]	[8] => [4, 9] (20%, 58%)	192.168.46.212
		[4, 8] => [9] (20%, 90%)	192.168.46.212
		[8, 9] => [4] (20%, 64%)	192.168.46.212
		[4] => [8, 9] (32%, 61%)	192.168.46.189
		[9] => [4, 8] (32%, 52%)	192.168.46.189
		[4, 8] => [9] (32%, 62%)	192.168.46.189
		[4, 9] => [8] (32%, 97%)	192.168.46.189
		[8, 9] => [4] (32%, 54%)	192.168.46.189
		[4] => [8, 9] (46%, 60%)	192.168.46.213
		[9] => [4, 8] (46%, 75%)	192.168.46.213
		[4, 8] => [9] (46%, 61%)	192.168.46.213
		[4, 9] => [8] (46%, 98%)	192.168.46.213
		[8, 9] => [4] (46%, 76%)	192.168.46.213
8	[7, 8, 9]	[8] => [7, 9] (22%, 64%)	192.168.46.212
		[7, 8] => [9] (22%, 90%)	192.168.46.212
		[8, 9] => [7] (22%, 71%)	192.168.46.212
		[7] => [8, 9] (41%, 61%)	192.168.46.189
		[9] => [7, 8] (41%, 65%)	192.168.46.189
		[7, 8] => [9] (41%, 63%)	192.168.46.189
		[7, 9] => [8] (41%, 96%)	192.168.46.189
		[8, 9] => [7] (41%, 68%)	192.168.46.189
		[7] => [8, 9] (29%, 59%)	192.168.46.213
		[7, 8] => [9] (29%, 60%)	192.168.46.213
		[7, 9] => [8] (29%, 97%)	192.168.46.213

Figure 4.7: Globally strong association rules for globally frequent 3-itemsets

Table 4.1: Comparative Analysis of AeMGSAR with Central DW based approach

Parameter	Analysis
Storage Cost	Size of L_k^{TFI} =12288 bytes(12KB) and Size of L^{TLSAR} = 176128 bytes (172KB). As the total size of data stored and managed at $S_{CENTRAL}$ in case of AeMGSAR is 188416 bytes (184KB) which is small as compared with the size (552KB) of central DB. Hence storage cost is reduced.
Communication Cost	Round Trip Time taken by TFICA in collecting L_k^{TFI} = 700441901 ns and Round Trip Time taken by LKCA in collecting L^{TLSAR} = 731541468 ns. Total time taken by TFICA and LKCA = 1431983369 ns which is 1252497636 ns less than the time taken in transferring DB_i. Hence communication cost is reduced in case of AeMGSAR.
Computational Cost	CPU time taken by GFIGA stationary agent = 101357102 ns and CPU time taken by GKGA stationary agent = 333174588 ns. Total CPU time taken by two major modules in Central DW based ARM is 509950438466 ns whereas total CPU time taken by two stationary agents at $S_{CENTRAL}$ is 434531690 ns which is 509515906776 ns less. Hence computation cost is reduced because this cost is shared locally by LFIGA and LKGA MAs.
Global Knowledge	AeMGSAR reflects the global knowledge because all the strong association rules generated are also strong at each distributed site, whereas Central DW based ARM does not reflect the global knowledge which also contains the strong rules for frequent 4-itemsets and are not globally frequent.
Security	DB_i is not transferred rather partial results are carried away by the MAs and the framework rely upon the Java's in-built security system . AeMGSAR framework is secure than its counterpart.
Scalability	As MAs are scalable in nature so performance would not be affected by adding more sites. Hence AeMGSAR framework is more scalable than its counterpart.

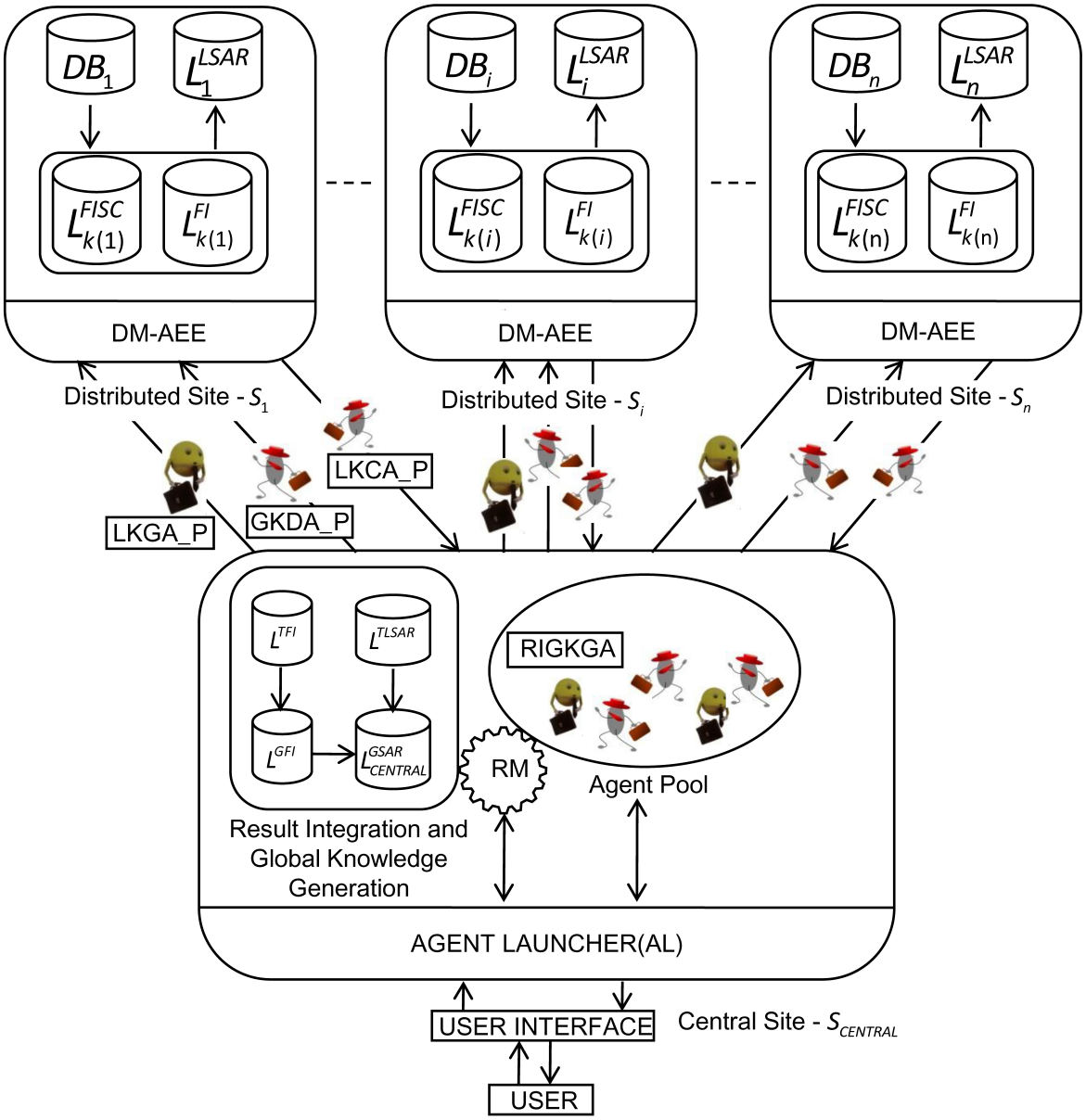

Figure 4.8: AeMGSAR Parallel Computing Model

4.3 Parallel Computing Model of AeMGSAR

Parallel computing model of AeMGSAR system is shown in Fig. 4.8. As compared to the serial computing model, it consists of total four agents, clones of three MAs, in serial number 1-3, are dispatched from $S_{CENTRAL}$ with parallel itinerary migration and one in serial number 4 is an intelligent SA running at $S_{CENTRAL}$ to perform different tasks. The CPU time taken by a MA while processing on each site along with some other specific information is carried back in the result bag at $S_{CENTRAL}$. Relationship among these agents and their working behavior are given below.

1. ***Local Knowledge Generater Agent with Parallel itinerary (LKGA_P)***: Every cloned $LKGA\_P$ carries the $AgentProfile$, min_th_sup and min_th_conf along with it. This agent is embedded with Apriori algorithm [5] for generating all the frequent k-itemset lists. This agent may be equipped with decision making capability to select other FIM algorithms based on the density of the dataset at a particular site. It first performs the FIM to generate and store $L^{FI}_{k(i)}$ and $L^{FISC}_{k(i)}$ at site S_i by scanning the local DB_i at that site with the constraint of min_th_sup. It makes use of a module described in Algorithm 5 in Chapter 3 to generate candidate frequent k-itemset list by joining a frequent (k-1)-itemset with itself. It performs the ARM applying the constraint of min_th_conf to generate and store L^{LSAR}_i by using the $L^{FI}_{k(i)}$ and $L^{FISC}_{k(i)}$ lists. L^{LSAR}_i list also contains support and confidence for a particular association rule along with the site name. It carries back the computational time ($CPUTime$) at each site S_i and $TripTime_{end}$. Steps involved are available in Algorithm 18.

2. ***Local Knowledge Collector Agent with Parallel itinerary (LKCA_P)***: Every cloned $LKCA\_P$ carries the $AgentProfile$ and collects the list of local frequent k-itemset ($L^{FI}_{k(i)}$) and list of locally strong association rules (L^{LSAR}_i) generated by $LKGA\_P$. It carries back these distributed results in the result bag to RM at $S_{CENTRAL}$ where these results are integrated with the help of $RIGKGA$ stationary agent. In addition to this resultant knowledge it also carries back the computational time ($CPUTime$) at each site S_i and $TripTime_{end}$. It executes Algorithm 19.

3. ***Global Knowledge Dispatcher Agent with Parallel itinerary (GKDA_P)***: Every cloned $GKDA\_P$ carries $AgentProfile$ containing global knowledge ($L^{GSAR}_{CENTRAL}$) for further decision making and comparing with the local knowledge at that site. It carries back the computational time ($CPUTime$) at each site S_i and $TripTime_{end}$. Details given in Algorithm 21.

4. ***Result Integration and Global Knowledge Generater Agent (RIGKGA)***: It is a stationary agent at $S_{CENTRAL}$, mainly used for processing the result bags of all clones of $LKCA\_P$. It creates a list of total frequent k-itemset (L^{TFI}_k), a list of global frequent itemset, L^{GFI}_k and a list of total locally strong association rules(L^{TLSAR}). L^{GFI}_k and L^{TLSAR} are further processed to generate and store $L^{GSAR}_{CENTRAL}$ list. Detailed steps are given in Algorithm 20.

Algorithm 18 Local Knowledge Generater Agent with Parallel itinerary (LKGA_P)

Input:

- $AgentProfile$, *A collection of Agent attributes set by the AL*
- min_th_sup, *the given minimum threshold support*
- min_th_conf, *the given minimum threshold confidence*

Output:

- $L^{FI\&SC}$, *the list of frequent itemsets and their support counts*
- L^{LSAR}, *the list of locally strong association rules*

1: **procedure** LKGA_P($AgentProfile, min\_th\_sup, min\_th\_conf$)
2: $CPUTime_{start} \leftarrow$ *get system time*
3: $Briefcase \leftarrow$ *get Briefcase from AgentProfile*
4: $DB_i \leftarrow$ *load* DB_i *from local file system of site* S_i
5: $T \leftarrow DB_i.get(0)$ ▷ No. of records
6: $I \leftarrow DB_i.get(1)$ ▷ No. of items
7: $DB[T][I] \leftarrow DB_i.get(3)$ ▷ itemset data bank
8: $minsupcount \leftarrow (T \times min\_th\_sup)/100$
9: ▷ generate frequent-1 itemset list (FIL_1) and support count list ($FISC_1$)
10: $CFIL_1 \leftarrow \{1, 2, 3 \cdots I\}$ ▷ candidate frequent-1 itemset
11: **for** $i \leftarrow 1, I$ **do** ▷ initialize the support count array $SCFIL_1$ to zero
12: $SCFIL_1[i] \leftarrow 0$
13: **end for**
14: $k \leftarrow 1$
15: **for all** *candidate* $c \in CFIL_1$ **do** ▷ find support count for every candidate
16: **for all** *transaction* $t \in DB$ **do** ▷ scan DB
17: **if** $c \subset t$ **then**
18: $SCFIL_1[k] \leftarrow SCFIL_1[k] + 1$
19: **end if**
20: **end for**
21: $k \leftarrow k + 1$
22: **end for**
23: ▷ prune $CFIL_1$ to generate FIL_1 and $FISC_1$
24: **for** $k \leftarrow 1, I$ **do**
25: **if** $SCFIL_1[k] \geq minsupcount$ **then**
26: *add* $c_k \in CFIL_1$ *to* FIL_1
27: *add* $SCFIL_1[k]$ *to* $FISC_1$
28: **end if**
29: **end for**
30: **if** $FIL_1 \neq \emptyset$ **then**
31: *add* FIL_1 *to* L^{FI} ▷ continue...

LKGA_P - continued

```
        add FISC_1 to L^FISC
    end if
    k ← 2
    while FIL_{k-1} ≠ ∅ do
        CFIL_k ← Call GenerateCFIL(FIL_{k-1})        ▷ see Algorithm 5 in chapter 3
        for i ← 1, CFIL_k.length do                  ▷ initialize the array SCFIL_k to zero
            SCFIL_k[i] ← 0
        end for
        i ← 1
        for all candidate c ∈ CFIL_k do              ▷ find support count for every candidate
            for all transaction t ∈ DB do                                  ▷ scan DB
                if c ⊂ t then
                    SCFIL_k[i] ← SCFIL_k[i] + 1
                end if
            end for
            i ← i + 1
        end for
        ▷ prune CFIL_k to generate FIL_k and FISC_k
        for i ← 1, SCFIL_k.length do
            if SCFIL_k[i] ≥ minsupcount then
                add c_i ∈ CFIL_k to FIL_k
                add SCFIL_k[i] to FISC_k
            end if
        end for
        if FIL_k ≠ ∅ then
            add FIL_k to L^FI
            add FISC_k to L^FISC
        end if
        k ← k + 1
    end while
    add T, L^FI and L^FISC to L^FI&SC
    save L^FI&SC in the local file system of this site S_i
    for k ← 2, L^FI.size do
        L_k ← L^FI.get(k)                            ▷ get frequent k-itemset list
        for all l ∈ L_k do
            l_subsets ← generate all non − empty subsets of l
            l_spcount ← get support count of l from L^FISC
            AR_support ← (l_spcount/T) × 100         ▷ support of the association rule
            for all non − empty subset s ∈ l_subsets do
                s_spcount ← get support count of s from L^FISC
                AR_conf ← (l_spcount/s_spcount) × 100   ▷ confidence of the association rule
                if AR_conf ≥ min_th_conf then
                    AR_strong ← "s ⇒ l − s[AR_support%, AR_conf%]"
                    print AR_strong                                  ▷ continue...
```

LKGA_P - continued

```
                    add l to AR_strong
                    S_i^IP ← get IP address of this site S_i
                    add S_i^IP to AR_strong
                    add AR_strong to L^LSAR
                end if
            end for
        end for
    end for
    save L^LSAR in the local file system of this site S_i
    CPUTime_end ← get system time
    CPUTime ← CPUTime_end − CPUTime_start
    add CPUTime to Result_S_i
    add Result_S_i to Briefcase
    TripTime_end ← get system time for end of agent journey
    add TripTime_end to Briefcase
    add updated Briefcase to AgentProfile
    transfer AgentProfile to RM at S_CENTRAL
end procedure
```

Algorithm 19 Local Knowledge Collector Agent with Parallel itinerary (LKCA_P)

Input: *AgentProfile, A collection of Agent attributes set by the Agent Launcher*
Output:

- L^{FI}, *the list of locally frequent itemsets*
- L^{LSAR}, *the list of locally strong association rules*

```
1:  procedure LKCA_P(AgentProfile)
2:      CPUTime_start ← get system time
3:      Briefcase ← get Briefcase from AgentProfile
4:      L^FI&SC ← load L^FI&SC from local file system of this site S_i
5:      L^FI ← L^FI&SC.get(1)                          ▷ frequent k-itemset list
6:      add L^FI to Result_S_i
7:      L^LSAR ← load L^LSAR from local file system of this site S_i
8:      add L^LSAR to Result_S_i
9:      CPUTime_end ← get system time
10:     CPUTime ← CPUTime_end − CPUTime_start
11:     add CPUTime to Result_S_i
12:     add Result_S_i to Briefcase
13:     TripTime_end ← get system time for end of agent journey
14:     add TripTime_end to Briefcase
15:     add updated Briefcase to AgentProfile
16:     transfer AgentProfile to RM at S_CENTRAL
17: end procedure
```

Algorithm 20 Result Integration and Global Knowledge Generater Agent (RIGKGA)

Input: $L_{AgentName}^{AllProfiles}$, *Result bags of all LKCA_P clones*
Output:

- L^{GFI}, *the list of global frequent itemsets*
- $L_{CENTRAL}^{GSAR}$, *the list of globally strong association rules*

1: **procedure** RIGKGA($L_{AgentName}^{AllProfiles}$)
2: $CPUTime_{start} \leftarrow$ *get system time*
3: $L^{TFI} \leftarrow$ *retrieve total frequent itemsets*$(\bigcup_{i=1}^{n} L_i^{FI})$ *from* $L_{AgentName}^{AllProfiles}$
4: $L^{GFI} \leftarrow$ *find global frequent itemsets*$(\bigcap_{i=1}^{n} L_i^{FI})$ *from* $L_{AgentName}^{AllProfiles}$
5: **print** L^{GFI}
6: $L^{TLSAR} \leftarrow$ *retrieve total strong rules*$(\bigcup_{i=1}^{n} L_i^{LSAR})$ *from* $L_{AgentName}^{AllProfiles}$
7: **for all** $AR_{strong} \in L^{TLSAR}$ **do**
8: $L \leftarrow$ *get frequent itemset from* AR_{strong}
9: **if** $L \in L^{GFI}$ **then**
10: **print** AR_{strong} *along with site address* (S_i^{IP})
11: *add* AR_{strong} *to* $L_{CENTRAL}^{GSAR}$
12: **end if**
13: **end for**
14: *save* L^{GFI} *and* $L_{CENTRAL}^{GSAR}$ *in the local file system of site* $S_{CENTRAL}$
15: $CPUTime_{end} \leftarrow$ *get system time*
16: $CPUTime \leftarrow CPUTime_{end} - CPUTime_{start}$
17: **print** $CPUTime$
18: **end procedure**

Algorithm 21 Global Knowledge Dispatcher Agent with Parallel itinerary (GKDA_P)

Input: *AgentProfile, A collection of Agent attributes set by the AL*
Output: *Dispatch* $L_{CENTRAL}^{GSAR}$ *at each distributed site* S_i

1: **procedure** GKDA_P(*AgentProfile*)
2: $CPUTime_{start} \leftarrow$ *get system time*
3: $Briefcase \leftarrow$ *get Briefcase from AgentProfile*
4: $L_{CENTRAL}^{GSAR} \leftarrow$ *get* $L_{CENTRAL}^{GSAR}$ *from Briefcase*
5: *save* $L_{CENTRAL}^{GSAR}$ *in the local file system of site* S_i
6: $CPUTime_{end} \leftarrow$ *get system time*
7: $CPUTime \leftarrow CPUTime_{end} - CPUTime_{start}$
8: *add CPUTime to Result_*S_i *add Result_*S_i *to Briefcase*
9: $TripTime_{end} \leftarrow$ *get system time for end of agent journey*
10: *add* $TripTime_{end}$ *to Briefcase*
11: *add updated Briefcase to AgentProfile*
12: *transfer AgentProfile to RM at* $S_{CENTRAL}$
13: **end procedure**

4.3.1 Overall Time Model

The overall response time T for the DARM task performed using parallel computing model of AeMGSAR system is calculated using the following equation:

$$T = t_{darm} + t_{kigkg} + t_{gkd} + t_{wait} \tag{4.13}$$

where,

- t_{darm} = total time taken for DARM task and is calculated using three estimates, of which first and third are communication costs and second one is computational cost:
  - Agent cost, which is the estimate of the maximum round trip time required for any MA to travel from $S_{CENTRAL}$ to S_i coming back at $S_{CENTRAL}$ form visited site, S_i.
  - Local ARM cost, which is the estimate of the maximum time needed for performing ARM task locally by clones of $LKGA\_P$ at each site S_i.
  - Local Knowledge Transfer cost, which is the estimate of the maximum round trip time needed by clones of $LKCA\_P$ to collect and submit results at $S_{CENTRAL}$.
- t_{kigkg} = time taken to perform distributed knowledge integration and global knowledge generation by $RIGKGA$ at $S_{CENTRAL}$.
- t_{gkd} = maximum round trip time taken by clones of $GKDA\_P$ for dispatching the global knowledge to distributed sites for local analysis.
- t_{wait} = total time period for which system will wait for an event to happen. It varies and depends on the availability of the active node.

4.3.1.1 Cost Model involving one Site

Let us now consider the case of i^{th} distributed site S_i where DM has to be performed. The t_{darm} component in equation 4.13 involving i^{th} distributed site S_i is calculated according to equation 4.14.

$$\begin{aligned} t_{darm} = t_{LKGA\_P}(S_{CENTRAL}, S_i) + t_{fim}(S_i) + t_{arm}(S_i) + t_{LKGA\_P}(S_i, S_{CENTRAL}) + \\ t_{LKCA\_P}(S_{CENTRAL}, S_i) + t_{LKCA\_P}(S_i, S_{CENTRAL}) \end{aligned} \tag{4.14}$$

where,

- $t_{LKGA\_P}(S_{CENTRAL}, S_i)$ = time taken for single $LKGA\_P$ (embedded with mining algorithm and other parameters like min_th_sup, min_th_conf) to move from $S_{CENTRAL}$ to i^{th} distributed site S_i.
- $t_{fim}(S_i)$ = time taken for performing FIM task locally by $LKGA\_P$ at S_i and is calculated according to equation 4.3
- $t_{arm}(S_i)$ = time taken for performing ARM task locally by $LKGA\_P$ at S_i and is calculated according to equation 4.4

- $t_{LKGA\_P}(S_i, S_{CENTRAL})$ = time taken by $LKGA\_P$ to move from S_i back to $S_{CENTRAL}$.
- $t_{LKCA\_P}(S_{CENTRAL}, S_i)$ = time taken by $LKCA\_P$ to move from $S_{CENTRAL}$ to S_i for collecting the desired result.
- $t_{LKCA\_P}(S_i, S_{CENTRAL})$ = time taken by $LKCA\_P$ to move from S_i back to $S_{CENTRAL}$ and submit the desired results to RM.

The t_{gkd} component in equation 4.13 is calculated using equation 4.15.

$$t_{gkd} = t_{GKDA\_P}(S_{CENTRAL}, S_i) + t_{GKDA\_P}(S_i, S_{CENTRAL}) \tag{4.15}$$

Putting equation 4.14 and 4.15 in equation 4.13, total response time T for DARM task for this case can be expressed as:

$$\begin{aligned} T = t_{LKGA\_P}(S_{CENTRAL}, S_i) + t_{fim}(S_i) + t_{arm}(S_i) + t_{LKGA\_P}(S_i, S_{CENTRAL}) + \\ t_{LKCA\_P}(S_{CENTRAL}, S_i) + t_{LKCA\_P}(S_i, S_{CENTRAL}) + t_{kigkg} + \\ t_{GKDA\_P}(S_{CENTRAL}, S_i) + t_{GKDA\_P}(S_i, S_{CENTRAL}) + t_{wait} \end{aligned} \tag{4.16}$$

In general, $t_{MobileAgent}(x, y)$ depends upon the size of the agent ($Size_{MobileAgent}$) and the network bandwidth (in Kbits per second) between two nodes x and y ($BWidth(x, y)$). Therefore,

$$t_{LKGA\_P}(S_{CENTRAL}, S_i) \propto \frac{k \times Size_{LKGA\_P}}{BWidth(S_{CENTRAL}, S_i)} \tag{4.17}$$

where k is a constant.

$$Size_{LKGA\_P} = \langle LKGA\_P\ State, FIM/ARM Algorithm, Input Parameters \rangle \tag{4.18}$$

$t_{LKGA\_P}(S_i, S_{CENTRAL})$, $t_{LKCA\_P}(S_{CENTRAL}, S_i)$, $t_{LKCA\_P}(S_i, S_{CENTRAL})$, $t_{GKDA\_P}(S_{CENTRAL}, S_i)$, and $t_{GKDA\_P}(S_i, S_{CENTRAL})$ is also calculated similar to $t_{LKGA\_P}(S_{CENTRAL}, S_i)$ and $Size_{LKCA\_P}$ does not include any DARM algorithm code and local knowledge is included as agent data component similarly global knowledge is included as agent data component in $Size_{GKDA\_P}$.

4.3.1.2 Cost Model involving n Distributed Sites

AL dispatches multiple cloned copy of each MA agent in parallel to n distributed sites. Mining is performed locally at each distributed site in parallel by $LKGA\_P$, then results are also transferred in parallel through $LKCA\_P$ to $S_{CENTRAL}$. Since mining is performed locally at each site simultaneously, the total time taken depends upon the time interval required by the site that takes longest time to perform DM and return results. Therefore,

$$\begin{aligned} t_{darm} = \max\left(t_{LKGA\_P}(S_{CENTRAL}, S_i) + t_{fim}(S_i) + t_{arm}(S_i) + t_{LKGA\_P}(S_i, S_{CENTRAL})\right) + \\ \max\left(t_{LKCA\_P}(S_{CENTRAL}, S_i) + t_{LKCA\_P}(S_i, S_{CENTRAL})\right), \forall i = \{1 \ldots n\} \end{aligned} \tag{4.19}$$

Putting equation 4.19 in equation 4.13 and taking the longest time to dispatch the global knowledge to distributed sites, overall response time T for the DARM using parallel computing model of AeMGSAR framework is given as:

$$T = \max\left(t_{LKGA\_P}(S_{CENTRAL}, S_i) + t_{fim}(S_i) + t_{arm}(S_i) + t_{LKGA\_P}(S_i, S_{CENTRAL})\right) + \max\left(t_{LKCA\_P}(S_{CENTRAL}, S_i) + t_{LKCA\_P}(S_i, S_{CENTRAL})\right) + t_{kigkg} + \max\left(t_{GKDA\_P}(S_{CENTRAL}, S_i) + t_{GKDA\_P}(S_i, S_{CENTRAL})\right) + t_{wait}, \forall i = \{1 \ldots n\} \quad (4.20)$$

Comparing equations 4.12 and 4.20 it is found that overall response time T for the DARM task involving n distributed sites is very less in case of parallel computing model of AeMGSAR as compared to the serial computing model of AeMGSAR framework.

4.4 Discussion

A comparative analysis of the proposed AeMGSAR framework is performed with three existing mobile agent based frameworks for DARM task. Table 4.2 shows the comparison of AeMGSAR with MADKDS [110]. Table 4.3 shows the comparison of AeMGSAR with AFARMDD [59] and finally Table 4.4 shows the comparison of AeMGSAR with MADARM [87].

Table 4.2: Comparison of AeMGSAR with MADKDS [110]

Parameters	**MADKDS [110]**	**AeMGSAR**
Runtime Environment at distributed sites	No algorithm and implementation of Mobile Agent Execution Environment module is discussed for receiving and handling various mobile agents involved.	An algorithm for Data Mining Agent Execution Environment module is designed and implemented in Java for receiving and handling various mobile agents involved (Algorithm 8).
Agent launching module at central site.	No algorithm and implementation of Mobile Agent Control Centre module is discussed for creating, launching and managing various agents involved.	An algorithm for Agent Launcher module is designed and implemented in Java for creating, launching and managing various agents involved (Algorithm 9).
Support count collection	A separate counter mobile agent is used to collect support count list of local frequent itemset.	The same mobile agent that collects the local frequent itemsets also takes support count list along with it. So cost of deploying additional mobile agent is reduced.

Parameters	MADKDS [110]	AeMGSAR
Data warehouse	A global data warehouse is maintained at central site that increases the storage cost at that site, network communication cost of transferring huge data from distributed sites.	No central data warehouse is maintained at central site and only resultant knowledge mined by mobile agents is maintained. Therefore it minimizes the storage and communication costs.
Global Knowledge Base	It consists of knowledge mined from the global data warehouse and the knowledge mined by agents.	It consists of knowledge mined by mobile agents and integrated by stationary agents at central site.
Synthetic dataset	No information about the synthetic dataset used for mining is provided.	A software tool called Transactional Dataset Generator was designed and implemented to create partitioned synthetic Boolean datasets at distributed sites.
Globally strong Rules	No information is given for the globally generated strong association rules.	List of globally frequent itemsets and globally strong rules are generated by Global Knowledge Generator Agent.
Cost Model of DARM task	No overall time model is discussed for the DARM task.	Overall time model is discussed for DARM task with serial (equation 4.1) and parallel (equation 4.13) computing model. Various factors affecting the cost model are also discussed.
Algorithms for Mobile agents	No algorithm is given for the working of any mobile agent and its itinerary management.	Algorithms for various stationary as well as mobile agents involved in both serial and parallel modals are designed and implemented in Java.

Table 4.3: Comparison of AeMGSAR with AFARMDD [59]

Parameters	**AFARMDD [59]**	**AeMGSAR**
Runtime Environment at distributed sites	No detailed algorithm and implementation of Local Host module is discussed for receiving and handling various mobile agents involved.	An algorithm for Data Mining Agent Execution Environment module is designed and implemented in Java for receiving and handling various mobile agents involved (Algorithm 8).
Agent launching module at central site.	Steps defined in algorithm for Agent Server module are unclear as agents are launched in parallel but they travel from node to node serially.	An algorithm for Agent Launcher module is precisely designed and implemented in Java for creating, launching and managing various agents involved (Algorithm 9).
FIM algorithm	It is not clear which component of the framework performs FIM to generate local frequent itemsets.	DM agent is embedded with Apriori [5] algorithm for mining the frequent itemsets from the local dataset.
Parallel itinerary	No steps are given in algorithms for agent for managing parallel itinerary.	Parallel computing modal is designed for the parallel itinerary of the mobile agents.
Global Knowledge Base	No information is given.	It consists of knowledge mined by mobile agents and integrated by stationary agents at central site.
Globally strong Rules	No information is given for the globally generated strong association rules.	List of globally frequent itemsets and globally strong rules are generated by Global Knowledge Generator Agent.
Cost Model of DARM task	No overall time model is discussed for the DARM task.	Overall time model is discussed for DARM task with serial (equation 4.1) and parallel (equation 4.13) computing modal. Various factors affecting the cost model are also discussed.
Algorithms for Mobile agents	Algorithms for the working of mobile agents are briefly discussed without itinerary management.	Algorithms for various stationary as well as mobile agents involved in both serial and parallel modals are designed and implemented in Java.
Computational and Trip time for agents	Round Trip time for the mobile agents and computational time at distributed and central site are not discussed.	Round Trip time and computational time for various agents involved are obtained in nano seconds.

Table 4.4: Comparison of AeMGSAR with MADARM [87]

Parameters	**MADARM [87]**	**AeMGSAR**
Runtime Environment at distributed sites	No algorithm and implementation of Mobile Agent Based Association Rule Miner module is discussed for receiving and handling various mobile agents involved.	An algorithm for Data Mining Agent Execution Environment module is designed and implemented in Java for receiving and handling various mobile agents involved (Algorithm 8).
Agent launching module at central site.	No algorithm and implementation of Association Rule Mining Coordinating Agent module is discussed for creating, launching and managing various agents involved.	An algorithm for Agent Launcher module is designed and implemented in Java for creating, launching and managing various agents involved (Algorithm 9).
Algorithms for Mobile agents	No algorithm is given for the working of any mobile agent and its itinerary management.	Algorithms for various stationary as well as mobile agents involved in both serial and parallel modals are designed and implemented in Java.
Synthetic dataset	No synthetic dataset is used for mining process.	A software tool called Transactional Dataset Generator was designed and implemented to create partitioned synthetic Boolean datasets at distributed sites.
Globally strong Rules	No information is given for the globally generated strong association rules.	List of globally frequent itemsets and globally strong rules are generated by Global Knowledge Generator Agent.
Cost Model of DARM task	The overall response time for the DARM task does not include time taken to perform global knowledge generation (t_{gkg}), time taken to dispatch global knowledge to distributed sites for local analysis (t_{gkd}) and total time period for which system will wait for an event to happen (t_{wait}). Various factors affecting the cost model are not taken into account.	Overall time model is discussed for DARM task with serial (equation 4.1) and parallel (equation 4.13) computing modal. Various factors affecting the cost model like size and density of the local dataset, number of other processes running at i^{th} site, the processor speed at i^{th} site and network bandwidth between two nodes are also discussed.
Computational and Trip time for agents	Round Trip time for the mobile agents and computational time at distributed and central site are not discussed.	Round Trip time and computational time for various agents involved are obtained in nano seconds.

The comparative analysis reveals that the proposed AeMGSAR framework has improved features and exhibit superior performance than the existing MADKDS [110], AFARMDD [59] and MADARM [87] frameworks for DARM task.

4.5 Summary

In this chapter agent based computing models are designed and implemented to mine the globally strong association rules from the distributed datasets. In these models, the overall task of mining the globally strong rules is divided into subtasks which are handled by various mobile as well as stationary agents. The performance of the serial computing model is compared with the central DW based approach. From the comparative analysis it is found that the storage, computation and communication costs are reduced in case of AeMGSAR serial computing model than central DW based ARM model. This serial modal deploys MAs with serial itinerary thereby increasing the overall time of the DARM task. An improved version of the framework AeMGSAR deploying cloned MAs with parallel itinerary is further proposed and designed that reduces the overall time of the DARM task. Qualitative comparison of AeMGSAR framework is performed with existing frameworks and found that AeMGSAR framework has improved features and exhibit superior performance than the existing framework for DARM task. As this MAS is tested on the synthetic datasets, the next chapter implements another MAS on the real protein data banks to find the association rules among amino acids.

Chapter 5

A CASE STUDY IN BIO-INFORMATICS

Cellular machinery of any organism comprises of proteins and functioning of proteins heavily depends upon the amino acid sequence present in it. The functioning of protein might completely change with a slight change in the amino acid sequence. Therefore, nature of associations between different amino acids has been a subject of great importance. Also the amalgamation of the DDM and MAS provides rewarding solution in terms of security, scalability, storage cost, computation cost and communication cost.

In this chapter, we have designed and implemented a MAS called Agent enriched Quantitative Association Rules Mining for Amino Acids in distributed Protein Data Banks (AeQARM-AAPDB). Such globally strong association rules enhance understanding of protein composition and are desirable for synthesis of artificial proteins. It is an attempt to bind the agent technology with DDM, particularly focusing on DARM emphasizing the use of intelligent agents in mining big biological datasets. Mining bio-informatics data is an emerging area at the intersection between bio-informatics and DM.

This chapter is organized as follows. Section 5.1 describes the proposed system AeQARM-AAPDB and protein data bank used. Working behaviour of various agents involved are discussed in Section 5.2. Section 5.3 describes the overall time model. Implementation, performance study and results validation are discussed in Section 5.4 and finally the chapter is summarized in Section 5.5.

5.1 AeQARM-AAPDB Multi Agent System

This MAS works on the top of underlying AEE already discussed in Section 4.1 in Chapter 4. All the components are same except AL that executes Algorithm 22 for this system. Basic working mechanism of AL is same but this time it handles the agents defined in Section 5.2. The working model of AeQARM-AAPDB is divided into seven stages which are as under.

1. ***Request Stage***: Request for DARM is initiated at $S_{CENTRAL}$ by AL on behalf of the user with necessary credentials.

2. ***Preparation Stage***: AL through User Interface reads agent name; version number; Itinerary (serial or parallel) for the MAs journey is obtained in terms of the IP

address of the distributed nodes to be visited by MA; any specific additional data for a specific MA is obtained; Agent code for the specific MA is loaded from Agent Pool; for serial itinerary a single specific MA is dispatched by AL to travel and visit n distributed sites; for parallel itinerary clones of a specific MA is dispatched by AL to visit n distributed sites in parallel.

3. ***Data filtering and Conversion Stage***: A data filtering agent filters the protein dataset with some criteria at each site and creates a filtered protein dataset. From such filtered datasets another agent finds the frequencies of each amino acids for each protein record and creates amino acids frequency dataset at each site; another agent processes this frequency dataset and divides the frequencies of each amino acids into intervals and maps the frequencies into boolean values to create boolean value dataset for frequency intervals and further maps this boolean value dataset to item dataset for frequency interval items.

4. ***Local Mining Stage***: ARM process is performed locally on item dataset by specific DM agents at each distributed site and results are kept as local knowledge base at that site.

5. ***Result Collection Stage***: Collector agents visit each site and collect the results generated by DM agents and submit the results back to RM at $S_{CENTRAL}$.

6. ***Knowledge Integration and Global Knowledge Generation Stage***: Knowledge or result integration is carried out by the RM with the help of stationary agents and Global Knowledge in the form of Globally Strong Association Rules may be generated with the help of other specific stationary agents at $S_{CENTRAL}$.

7. ***Global Knowledge Dispatching Stage***: Global Knowledge is dispatched to the distributed sites by a dispatcher agent to compare it with the local knowledge at each site.

5.1.1 Preliminaries and Definitions

Preliminaries notations used in the framework are already discussed in Section 1.3.3.1 of Chapter 2. In addition to these following notations are also used in the framework.

- $PDB_i = \{PR_m, m = 1 \cdots X_i\}$, Protein Data Bank of X_i protein records at site S_i, each PR_m has two main parts; first part is the description headers (PD) for structural classification of protein and the second part contains protein sequence (PS), i.e., the chain of amino acids. PDB_1, PDB_2 and PDB_3 are shown in Appendix C.1, C.2 and C.3 respectively.

- $FPDB_i = \{PR_k, k = 1 \cdots D_i\}$, Filtered Protein Data Bank of D_i protein records at site S_i, where length of PS in each PR is in the range ≥ 50 and < 400. $FPDB_1$, $FPDB_2$ and $FPDB_3$ are shown in Appendix C.4, C.5 and C.6, respectively.

- AAF_i, Data bank of amino acids frequency for each $PR \in FPDB$ at site S_i. AAF_1, AAF_2 and AAF_3 are shown in Appendix C.7, C.8 and C.9, respectively.

- BDB_i, Boolean Data Bank that contains a value '1' if the frequency of an amino acid lies in the specific range and '0' otherwise at site S_i. BDB_1, BDB_2 and BDB_3 are shown in Appendix C.11, C.12 and C.13, respectively.
- IDB_i, Itemset Data Bank of frequency partition items to map boolean value '1 with its frequency partition item number at site S_i. IDB_1, IDB_2 and IDB_3 are shown in Appendix C.14, C.15 and C.16, respectively.

Algorithm 22 AGENT LAUNCHER(AL) FOR AEQARM-AAPDB

```
1: procedure AL( )
2:     option ← read option(dispatch/result)
3:     switch option do
4:         case dispatch                          ▷ dispatch the mobile agent to DM_AEE
5:             AgentName ← read Mobile Agent's name
6:             BC ← load entire BC of AgentName from AgentPool
7:             add AgentName and BC to AgentProfile
8:             L^NODES ← read Itinerary(IP addresses) of mobile agent
9:             ItinType ← read ItineraryType(Serial/Parallel)
10:            add ItinType to AgentProfile
11:            if ItinType = "Parallel" then                    ▷ Parallel Itinerary
12:                AObject[L^NODES.size]   ▷ Array of Agent Objects for clone references
13:                TripTime_start ← get system time for start of agent journey
14:                add TripTime_start to Briefcase
15:                add Briefcase to AgentProfile
16:                switch AgantName do
17:                    case PDBFA
18:                        for i ← 1, L^NODES.size do      ▷ for each node in the itinerary
19:                            AgentVersion ← i
20:                            add AgentVersion to AgentProfile
21:                            NodeAddress ← L^NODES.get(i)          ▷ get an IP address
22:                            AObject[i] ← new PDBFA(AgentProfile)
23:                            Add AObject[i] to AgentProfile          ▷ clone's state
24:                            Transfer AgentProfile to DM_AEE at NodeAddress
25:                        end for
26:                    end case
27:                    case AAFFA
28:                        for i ← 1, L^NODES.size do      ▷ for each node in the itinerary
29:                            AgentVersion ← i
30:                            add AgentVersion to AgentProfile
31:                            NodeAddress ← L^NODES.get(i)          ▷ get an IP address
32:                            AObject[i] ← new AAFFA(AgentProfile)      ▷ continue...
```

AL - continued

```
33:                 Add AObject[i] to AgentProfile                ▷ clone's state
34:                 Transfer AgentProfile to DM_AEE at NodeAddress
35:             end for
36:         end case
37:         case FMIDBGA
38:             max_freq ← read maximum frequency range for amino acids
39:             for i ← 1, L^NODES.size do
40:                 AgentVersion ← i
41:                 add AgentVersion to AgentProfile
42:                 NodeAddress ← L^NODES.get(i)
43:                 AObject[i] ← new FMIDBGA(AgentProfile, max_freq)
44:                 Add AObject[i] to AgentProfile                ▷ clone's state
45:                 Transfer AgentProfile to DM_AEE at NodeAddress
46:             end for
47:         end case
48:         case LKGA_P
49:             min_s ← read minimum threshold support
50:             min_c ← read minimum threshold confidence
51:             for i ← 1, L^NODES.size do        ▷ for each node in the itinerary
52:                 AgentVersion ← i
53:                 add AgentVersion to AgentProfile
54:                 NodeAddress ← L^NODES.get(i)              ▷ get an IP address
55:                 AObject[i] ← new LKGA_P(AgentProfile, min_s, min_c)
56:                 Add AObject[i] to AgentProfile                ▷ clone's state
57:                 Transfer AgentProfile to DM_AEE at NodeAddress
58:             end for
59:         end case
60:         case LKCA_P
61:             for i ← 1, L^NODES.size do        ▷ for each node in the itinerary
62:                 AgentVersion ← i
63:                 add AgentVersion to AgentProfile
64:                 NodeAddress ← L^NODES.get(i)              ▷ get an IP address
65:                 AObject[i] ← new LKCA_P(AgentProfile)
66:                 Add AObject[i] to AgentProfile                ▷ clone's state
67:                 Transfer AgentProfile to DM_AEE at NodeAddress
68:             end for
69:         end case
70:         case GKDA_P
71:             L^GSAR_CENTRAL ← load L^GSAR_CENTRAL generated by GKGA at S_CENTRAL
72:             add L^GSAR_CENTRAL to Briefcase
73:             add updated Briefcase to AgentProfile
74:             for i ← 1, L^NODES.size do        ▷ for each node in the itinerary
75:                 AgentVersion ← i
76:                 add AgentVersion to AgentProfile
77:                 NodeAddress ← L^NODES.get(i)              ▷ get an IP address
78:                 AObject[i] ← new GKDA_P(AgentProfile)     ▷ continue...
```

AL - continued

```
79:                         Add AObject[i] to AgentProfile                ▷ clone's state
80:                         Transfer AgentProfile to DM_AEE at NodeAddress
81:                     end for
82:                 end case
83:             end switch
84:         end if
85:     end case
86:     case result                                  ▷ process the results of mobile agent
87:         AgentName ← read mobile agent's name
88:         ItinType ← read mobile agent's Itinerary Type
89:         add AgentName and ItinType to L^AgentInfo
90:         if ItinType = "Parallel" then
91:             L^AllProfile_AgentName ← contact RM for L^AgentInfo
92:             switch AgentName do
93:                 case PDBFA
94:                     for all AgentProfile ∈ L^AllProfile_AgentName do      ▷ for each clone
95:                         Briefcase ← retrieve Briefcase from AgentProfile
96:                         process the Briefcase of PDBFA clone
97:                     end for
98:                 end case
99:                 case AAFFA
100:                    for all AgentProfile ∈ L^AllProfile_AgentName do      ▷ for each clone
101:                        Briefcase ← retrieve Briefcase from AgentProfile
102:                        process the Briefcase of AAFFA clone
103:                    end for
104:                end case
105:                case FMIDBGA
106:                    for all AgentProfile ∈ L^AllProfile_AgentName do      ▷ for each clone
107:                        Briefcase ← retrieve Briefcase from AgentProfile
108:                        process the Briefcase of FMIDBGA clone
109:                    end for
110:                end case
111:                case LKGA_P
112:                    for all AgentProfile ∈ L^AllProfile_AgentName do      ▷ for each clone
113:                        Briefcase ← retrieve Briefcase from AgentProfile
114:                        process the Briefcase of LKGA_P clone
115:                    end for
116:                end case
117:                case LKCA_P
118:                    call RIGKGA(L^AllProfile_AgentName)                   ▷ stationary agent
119:                end case
120:                case GKDA_P
121:                    for all AgentProfile ∈ L^AllProfile_AgentName do      ▷ for each clone
122:                        Briefcase ← retrieve Briefcase from AgentProfile
123:                        process the Briefcase of GKDA_P clone
124:                    end for                                               ▷ continue...
```

AL - continued

```
                    end case
                end switch
            end if
        end case
    end switch
end procedure
```

5.1.2 Protein Data Bank

AeQARM-AAPDB system is tested on a real dataset of protein sequences, the Astral SCOP [75, 84] version 1.75 genetic domain sequence subsets, based on PDB SEQRES records with less than 40% identity to each other [12]. There are total of 10569 Protein records in this dataset. This single PDB of 10569 records is divided into 3 units (PDB_1, PDB_2 and PDB_3) of 3523 protein records in each and are stored at three distributed sites. Each PDB_i is further filtered to generate $FPDB_i$ for the PS length range ≥ 50 and < 400. A total of only 9633 (3341 ($FPDB_1$) + 3253 ($FPDB_2$) + 3039 ($FPDB_3$)) such filtered protein records are considered for the global mining. The amino acids frequencies in each protein sequence are retrieved and stored in AAF_i which is further mapped into BDB_i for each amino acid having 15 frequency ranges (partitions) resulting into 300 amino acids $\langle attribute, value \rangle$ pairs as shown in Appendix C.10.

5.2 Layout and working of AeQARM-AAPDB

AeQARM-AAPDB MAS is shown in Fig. 5.1. This MAS consists of total seven agents, clones of six MAs in serial number 1 to 6 are dispatched from $S_{CENTRAL}$ with parallel itinerary migration and one at serial number 7 is an intelligent SA running at $S_{CENTRAL}$ to perform different tasks. The CPU time taken by a MA while processing on each site along with some other specific information is carried back in the result bag at $S_{CENTRAL}$. Relationship among these agents and their working behaviour are given below.

1. ***Protein Data Bank Filtering Agent (PDBFA)***: Clones of this MA is dispatched in parallel to each distributed site by AL. It carries the $AgentProfile$ along with it and filters PDB_i to generate $FPDB_i$ at each site S_i. $PDBFA$ carries back the computational time ($CPUTime$) at each site S_i and $TripTime_{end}$. Detailed steps are given in Algorithm 23.

2. ***Amino Acids Frequency Finder Agent (AAFFA)***: Every clone of this MA carries the $AgentProfile$ along with it and finds the frequencies of each amino acids in every protein sequence record in $FPDB_i$ to create AAF_i at each site S_i. $AAFFA$ carries back the computational time ($CPUTime$) at each site S_i and $TripTime_{end}$. Steps are available in Algorithm 24.

3. ***Frequency Mapping and Itemset Data Bank Generater Agent (FMIDBGA)***: Every clone of this MA carries the $AgentProfile$ and $maxfrq$ (the given maximum frequency range for amino acids) along with it. It divides the frequencies

of each amino acids in AAF_i into intervals and maps the frequencies into boolean values to create BDB_i for frequency intervals and further maps it to IDB_i for frequency interval items. It executes Algorithm 25.

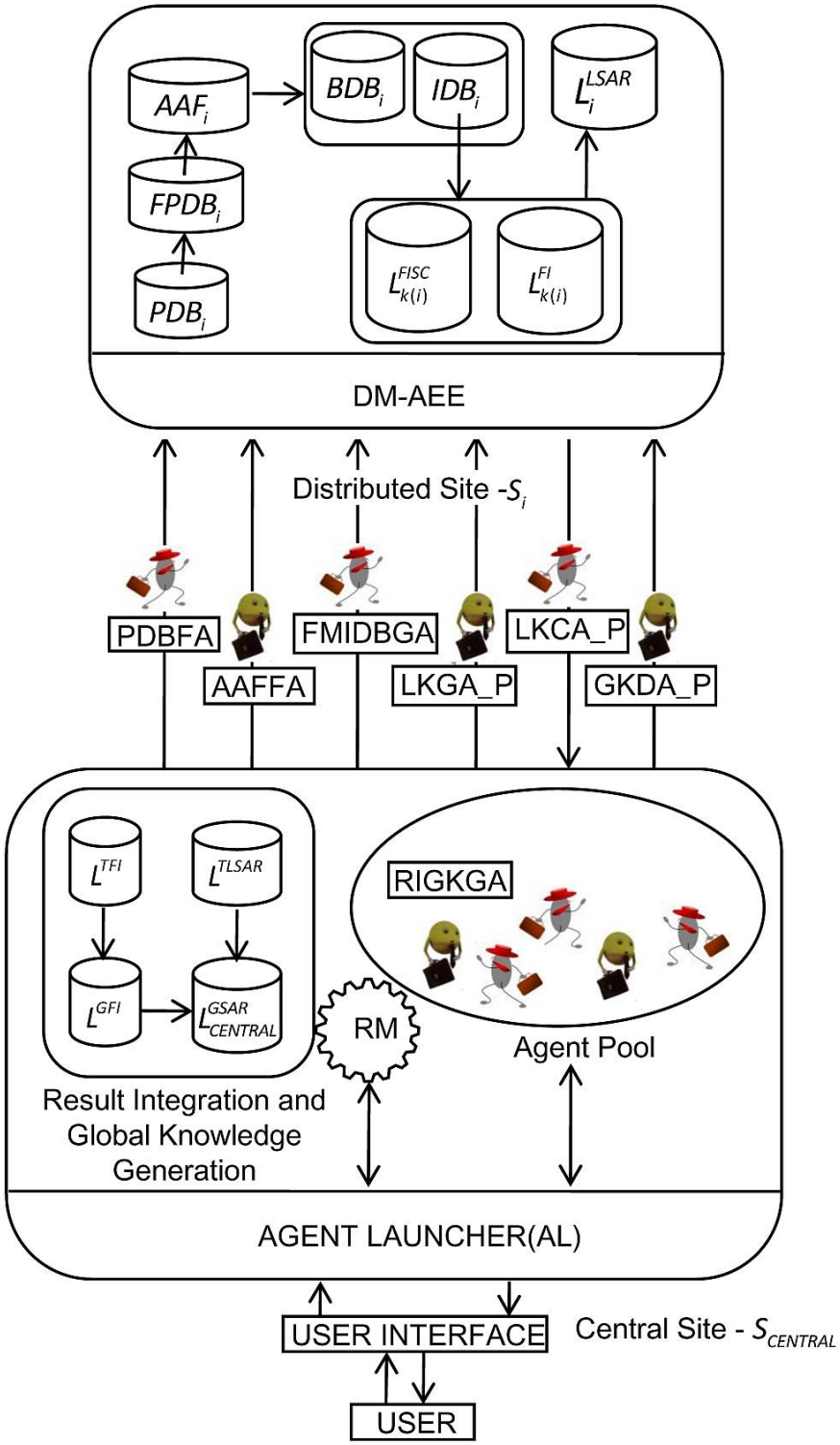

Figure 5.1: AeQARM-AAPDB MAS

4. ***Local Knowledge Generater Agent with Parallel itinerary (LKGA_P)***: It is similar to the component discussed in Section 4.3 in Chapter 4. Instead of loading DB_i at Step 4 in Algorithm 18, now it loads IDB_i from the local file system of site S_i.

5. ***Local Knowledge Collector Agent with Parallel itinerary (LKCA_P)***:It is also similar to the component discussed in Section 4.3 in Chapter 4 and executes Algorithm 19.

6. ***Global Knowledge Dispatcher Agent with Parallel itinerary (GKDA_P)***: It is also similar to the component discussed in Section 4.3 in Chapter 4 and executes Algorithm 21.

7. ***Result Integration and Global Knowledge Generater Agent (RIGKGA)***: It is also similar to the component discussed in Section 4.3 in Chapter 4 and executes Algorithm 20.

Algorithm 23 PROTEIN DATA BANK FILTERING AGENT (PDBFA)

Input: *AgentProfile, A collection of Agent attributes set by the AL*
Output: $FPDB_i$, *Filtered protein data bank at site* S_i

1: **procedure** PDBFA(*AgentProfile*)
2: $CPUTime_{start} \leftarrow$ *get system time*
3: $Briefcase \leftarrow$ *get Briefcase from AgentProfile*
4: $PDB_i \leftarrow$ *load* PDB_i *from local file system of this site* S_i
5: **for all** $PR \in PDB_i$ **do**
6: $PD \leftarrow$ *extract protein description headers from PR*
7: $PS \leftarrow$ *extract protein sequence from PR*
8: **if** $PS.length \geq 50$ *AND* $PS.length < 400$ **then**
9: *add PD to* L^{PD}
10: *add PS to* L^{PS}
11: **end if**
12: **end for**
13: *add* L^{PD} *to* $FPDB_i$
14: *add* L^{PS} *to* $FPDB_i$
15: *save* $FPDB_i$ *in the local file system of this site* S_i
16: $CPUTime_{end} \leftarrow$ *get system time*
17: $CPUTime \leftarrow CPUTime_{end} - CPUTime_{start}$
18: *add CPUTime to* $Result\_S_i$
19: *add* $Result\_S_i$ *to Briefcase*
20: $TripTime_{end} \leftarrow$ *get system time for end of agent journey*
21: *add* $TripTime_{end}$ *to Briefcase*
22: *add updated Briefcase to AgentProfile*
23: *transfer AgentProfile to RM at* $S_{CENTRAL}$
24: **end procedure**

Algorithm 24 Amino Acids Frequency Finder Agent (AAFFA)

Input: *AgentProfile, A collection of Agent attributes set by the AL*
Output: AAF_i, *Amino acid frequency data bank at site* S_i

```
1: procedure AAFFA(AgentProfile)
2:     CPUTime_start ← get system time
3:     Briefcase ← get Briefcase from AgentProfile
4:     FPDB_i ← load FPDB_i from local file system of this site S_i
5:     L^PD ← FPDB_i.get(0)
6:     L^PS ← FPDB_i.get(1)
7:     AAF[L^PS.size][20]                    ▷ array to store amino acids frequencies
8:     AACodes ← "acdefghiklmnpqrstvwy"      ▷ 20 single letter amino acid codes
9:     for ps ← 0, L^PS.size do
10:        PS ← L^PS.get(ps)                 ▷ retrieve protein sequence
11:        for aac ← 0, AACodes.length do    ▷ for each amino acid
12:            AA ← AACodes.charAt(aac)
13:            freq ← 0
14:            for psc ← 0, PS.length do
15:                PSAA ← PS.charAt(psc)
16:                if AA = PSAA then
17:                    freq ← freq + 1
18:                end if
19:            end for
20:            AAF[ps][aac] ← freq
21:        end for
22:    end for
23:    add L^PD to AAF_i
24:    add AAF[L^PS.size][20] to AAF_i
25:    save AAF_i in the local file system of this site S_i
26:    CPUTime_end ← get system time
27:    CPUTime ← CPUTime_end − CPUTime_start
28:    add CPUTime to Result_S_i
29:    add Result_S_i to Briefcase
30:    TripTime_end ← get system time for end of agent journey
31:    add TripTime_end to Briefcase
32:    add updated Briefcase to AgentProfile
33:    transfer AgentProfile to RM at S_CENTRAL
34: end procedure
```

Algorithm 25 Frequency Mapping and Itemset Data Bank Generater Agent (FMIDBGA)

Input:

- $AgentProfile$, *A collection of Agent attributes set by the AL*
- $maxfrq$, *the given maximum frequency range for amino acids*

Output: IDB_i, *Itemset data bank at site* S_i

```
1: procedure FMIDBGA(AgentProfile, maxfrq)
2:     CPUTime_start ← get system time
3:     Briefcase ← get Briefcase from AgentProfile
4:     AAF_i ← load AAF_i from local file system of this site S_i
5:     L^PD ← AAF_i.get(0)
6:     AAF[L^PD.size()][20] ← AAF_i.get(1)          ▷ to store AA frequencies
7:     ▷map AAF to BDB
8:     TempBDB ← new int[20][L^PD.size()][15]       ▷ temporary 3-D integer array
9:     for pr ← 0, AAF_i.length − 1 do              ▷ for each protein record
10:        for aa ← 0, 19 do                        ▷ for each of 20 amino acid
11:            freq ← AAF[pr][aa]
12:            if freq ≥ 0 AND freq ≤ 2 then
13:                fr ← 0
14:            end if
15:            if freq ≥ 3 AND freq ≤ 5 then
16:                fr ← 1
17:            end if
18:            if freq ≥ 6 AND freq ≤ 8 then
19:                fr ← 2
20:            end if
21:            if freq ≥ 9 AND freq ≤ 11 then
22:                fr ← 3
23:            end if
24:            if freq ≥ 12 AND freq ≤ 14 then
25:                fr ← 4
26:            end if
27:            if freq ≥ 15 AND freq ≤ 17 then
28:                fr ← 5
29:            end if
30:            if freq ≥ 18 AND freq ≤ 20 then
31:                fr ← 6
32:            end if
33:            if freq ≥ 21 AND freq ≤ 30 then
34:                fr ← 7
35:            end if
36:            if freq ≥ 31 AND freq ≤ 40 then
37:                fr ← 8
38:            end if                               ▷ continue...
```

FMIDBGA - continued

```
            if freq ≥ 41 AND freq ≤ 50 then
                fr ← 9
            end if
            if freq ≥ 51 AND freq ≤ 60 then
                fr ← 10
            end if
            if freq ≥ 61 AND freq ≤ 70 then
                fr ← 11
            end if
            if freq ≥ 71 AND freq ≤ 80 then
                fr ← 12
            end if
            if freq ≥ 81 AND freq ≤ 90 then
                fr ← 13
            end if
            if freq ≥ 91 AND freq ≤ maxfrq then
                fr ← 14
            end if
            TempBDB[aa][pr][fr] ← 1                    ▷ store boolean value '1'
        end for
    end for
    BDB ← new int[L^PD.size()][300]                    ▷ Boolean data bank
    ctr ← 0
    for aa ← 0, 19 do                                  ▷ for each 20 amino acids
        AABF ← TempBDB[aa]
        for fr ← 0, 14 do                              ▷ for each column of AABF array
            for pr ← 0, AABF.length − 1 do             ▷ for each protein record
                BDB[pr][ctr] ← AABF[pr][fr]
            end for
            ctr ← ctr + 1
        end for
    end for
    ▷ map BDB to IDB
    IDB ← new int[L^PD.size()][02]                     ▷ itemset data bank
    for i ← 0, L^PD.size() − 1 do
        k ← 0
        for j ← 0, 299 do
            if BDB[i][j] = 1 then
                IDB[i][k] ← j + 1
                k ← k + 1
            end if
        end for
    end for
    T ← L^PD.size()                                    ▷ no. of records
    I ← 300                                            ▷ no. of items
    add T to IDB_i                                     ▷ continue...
```

FMIDBGA - continued

85: *add* I *to* IDB_i ▷ no. of items
86: *add* L^{PD} *to* IDB_i
87: *add* $IDB[L^{PD}.size()][20]$ *to* IDB_i
88: *save* IDB_i *in the local file system of this site* S_i
89: $CPUTime_{end} \leftarrow$ *get system time*
90: $CPUTime \leftarrow CPUTime_{end} - CPUTime_{start}$
91: *add* $CPUTime$ *to* $Result\_S_i$
92: *add* $Result\_S_i$ *to* $Briefcase$
93: $TripTime_{end} \leftarrow$ *get system time for end of agent journey*
94: *add* $TripTime_{end}$ *to* $Briefcase$
95: *add updated* $Briefcase$ *to* $AgentProfile$
96: *transfer* $AgentProfile$ *to RM at* $S_{CENTRAL}$
97: **end procedure**

5.3 Overall Time Model

The overall response time T for the DARM performed using AeQARM-AAPDB system is calculated using the following equation:

$$T = t_{darm} + t_{kigkg} + t_{gkd} + t_{wait} \tag{5.1}$$

where,

- t_{darm} = total time taken for DARM task and is calculated using four estimates, of which first & forth are communication costs and second & third are the computational cost:
  - Agent cost, which is the estimate of the maximum round trip time required for any MA to travel from $S_{CENTRAL}$ to S_i coming back at $S_{CENTRAL}$ from visited site, S_i.
  - Local Data Filtering and Conversion cost, which is the estimate of the time needed for performing data filtering and boolean conversion task locally by *PDBFA*, *AAFFA* and *FMIDBGA* agents at each site S_i.
  - Local ARM cost, which is the estimate of the maximum time needed for performing ARM task locally by clones of *LKGA_P* at each site S_i.
  - Local Knowledge Transfer cost, which is the estimate of the maximum round trip time needed by clones of *LKCA_P* to collect and submit results to the $S_{CENTRAL}$.
- t_{kigkg} = time taken to perform distributed knowledge integration and global knowledge generation by *RIGKGA* at $S_{CENTRAL}$.
- t_{gkd} = maximum round trip time taken by clones of *GKDA_P* for dispatching the global knowledge to distributed sites for local analysis.

- t_{wait} = total time period for which system will wait for an event to happen. It varies and depends on the availability of the active node.

5.3.1 Cost Model involving one Site

Let us now consider the case of i^{th} distributed site S_i where DM has to be performed. Let time taken by a MA to travel from node x to node y is refered by $t_{MobileAgent}(x, y)$. The t_{darm} component in equation 5.1 involving i^{th} distributed site S_i is calculated according to equation 5.2.

$$\begin{aligned} t_{darm} = {} & t_{PDBFA}(S_{CENTRAL}, S_i) + t_{filtering}(S_i) + t_{PDBFA}(S_i, S_{CENTRAL}) + \\ & t_{AAFFA}(S_{CENTRAL}, S_i) + t_{frequency}(S_i) + t_{AAFFA}(S_i, S_{CENTRAL}) + \\ & t_{FMIDBGA}(S_{CENTRAL}, S_i) + t_{conversion}(S_i) + t_{FMIDBGA}(S_i, S_{CENTRAL}) + \\ & t_{LKGA\_P}(S_{CENTRAL}, S_i) + t_{fim}(S_i) + t_{arm}(S_i) + t_{LKGA\_P}(S_i, S_{CENTRAL}) + \\ & t_{LKCA\_P}(S_{CENTRAL}, S_i) + t_{LKCA\_P}(S_i, S_{CENTRAL}) \end{aligned} \tag{5.2}$$

where,

- $t_{PDBFA}(S_{CENTRAL}, S_i)$ = time taken for *PDBFA* to move from $S_{CENTRAL}$ to i^{th} distributed site S_i.
- $t_{filtering}(S_i)$ = time taken for data filtering task performed locally by *PDBFA* at S_i to generate $FPDB_i$ and it depends upon the following factors: the size of protein data bank ($Size\,(PDB_i)$), no. of other processes running at the local site ($\#\_of\_other\_Processes\,(S_i)$) and the processors speed at S_i ($P\_Speed\,(S_i)$). Therefore,

  $$t_{filtering}(S_i) \propto \frac{k \times Size(PDB_i) \times \#\_of\_other\_Processes(S_i)}{P\_Speed(S_i)} \tag{5.3}$$

  where, k is a constant.
- $t_{PDBFA}(S_i, S_{CENTRAL})$ = time taken by *PDBFA* to move from S_i back to $S_{CENTRAL}$.
- $t_{AAFFA}(S_{CENTRAL}, S_i)$ = time taken by *AAFFA* to move from $S_{CENTRAL}$ to S_i.
- $t_{frequency}(S_i)$ = time taken for amino acid frequency finding task performed locally by *AAFFA* at S_i to generate AAF_i and it depends upon the following factors: the size of filtered protein data bank ($Size\,(FPDB_i)$), $\#\_of\_other\_Processes\,(S_i)$) and $P\_Speed\,(S_i)$. Therefore,

  $$t_{frequency}(S_i) \propto \frac{k \times Size(FPDB_i) \times \#\_of\_other\_Processes(S_i)}{P\_Speed(S_i)} \tag{5.4}$$

  where, k is a constant.
- $t_{AAFFA}(S_i, S_{CENTRAL})$ = time taken by *AAFFA* to move from S_i back to $S_{CENTRAL}$.
- $t_{FMIDBGA}(S_{CENTRAL}, S_i)$ = time taken by *FMIDBGA* to move from $S_{CENTRAL}$ to S_i.

- $t_{conversion}(S_i)$ = time taken for mapping of amino acid frequency into boolean values (BDB_i) and then converting these boolean values to itemset data bank (IDB_i) locally by *FMIDBGA* at S_i and it depends upon the following factors: the size of amino acids frequency array ($Size\,(AAF_i)$), no. of frequency ranges for a single amino acids(FR_{AA}), $\#\_of\_other\_Processes\,(S_i)$ and $P\_Speed\,(S_i)$. Therefore,

$$t_{conversion}(S_i) \propto \frac{k \times Size(AAF_i) \times FR_{AA} \times \#\_of\_other\_Processes(S_i)}{P\_Speed(S_i)} \tag{5.5}$$

  where, k is a constant.

- $t_{FMIDBGA}(S_i, S_{CENTRAL})$ = time taken by *FMIDBGA* to move from S_i back to $S_{CENTRAL}$.

- $t_{LKGA\_P}(S_{CENTRAL}, S_i)$ = time taken by $LKGA\_P$(embedded with mining algorithm and other parameters like min_th_sup, min_th_conf) to move from $S_{CENTRAL}$ to S_i.

- $t_{fim}(S_i)$ = time taken for performing FIM task locally by $LKGA\_P$ at S_i and is calculated according to equation 4.3 in chapter 4

- $t_{arm}(S_i)$ = time taken for performing ARM task locally by $LKGA\_P$ at S_i and is calculated according to equation 4.4 in chapter 4

- $t_{LKGA\_P}(S_i, S_{CENTRAL})$ = time taken by $LKGA\_P$ to move from S_i back to $S_{CENTRAL}$.

- $t_{LKCA\_P}(S_{CENTRAL}, S_i)$ = time taken by $LKCA\_P$ to move from $S_{CENTRAL}$ to S_i to collect the desired result.

- $t_{LKCA\_P}(S_i, S_{CENTRAL})$ = time taken by $LKCA\_P$ to move from S_i back to $S_{CENTRAL}$ and submit the desired results to *RM*.

The t_{gkd} component in equation 5.1 is calculated using equation 5.6.

$$t_{gkd} = t_{GKDA\_P}(S_{CENTRAL}, S_i) + t_{GKDA\_P}(S_i, S_{CENTRAL}) \tag{5.6}$$

Putting equation 5.2 and equation 5.6 in equation 5.1, total response time T for DARM task for this case can be expressed as:

$$\begin{aligned} T = {} & t_{PDBFA}(S_{CENTRAL}, S_i) + t_{filtering}(S_i) + t_{PDBFA}(S_i, S_{CENTRAL}) + \\ & t_{AAFFA}(S_{CENTRAL}, S_i) + t_{frequency}(S_i) + t_{AAFFA}(S_i, S_{CENTRAL}) + \\ & t_{FMIDBGA}(S_{CENTRAL}, S_i) + t_{conversion}(S_i) + t_{FMIDBGA}(S_i, S_{CENTRAL}) + \\ & t_{LKGA\_P}(S_{CENTRAL}, S_i) + t_{fim}(S_i) + t_{arm}(S_i) + t_{LKGA\_P}(S_i, S_{CENTRAL}) + \\ & t_{LKCA\_P}(S_{CENTRAL}, S_i) + t_{LKCA\_P}(S_i, S_{CENTRAL}) + t_{kigkg} + \\ & t_{GKDA\_P}(S_{CENTRAL}, S_i) + t_{GKDA\_P}(S_i, S_{CENTRAL}) + t_{wait} \end{aligned} \tag{5.7}$$

In general, $t_{MobileAgent}(x, y)$ depends upon the size of the agent ($Size_{MobileAgent}$) and the network bandwidth (in Kbits per second) between two nodes x and y ($BWidth(x, y)$).

Therefore,

$$t_{LKGA\_P}(S_{CENTRAL}, S_i) \propto \frac{k \times Size_{LKGA\_P}}{BWidth(S_{CENTRAL}, S_i)} \tag{5.8}$$

where k is a constant.

$$Size_{LKGA\_P} = \langle LKGA\_P\ State, FIM/ARM Algorithm, Input Parameters \rangle \tag{5.9}$$

$t_{LKGA\_P}(S_i, S_{CENTRAL})$, $t_{PDBFA}(S_{CENTRAL}, S_i)$, $t_{PDBFA}(S_i, S_{CENTRAL})$, $t_{AAFFA}(S_{CENTRAL}, S_i)$, $t_{AAFFA}(S_i, S_{CENTRAL})$, $t_{FMIDBGA}(S_{CENTRAL}, S_i)$, $t_{AAFFA}(S_i, S_{CENTRAL})$, $t_{LKCA\_P}(S_{CENTRAL}, S_i)$, $t_{LKCA\_P}(S_i, S_{CENTRAL})$, $t_{GKDA\_P}(S_{CENTRAL}, S_i)$, and $t_{GKDA\_P}(S_i, S_{CENTRAL})$ is also calculated similar to $t_{LKGA\_P}(S_{CENTRAL}, S_i)$ and $Size_{LKCA\_P}$ does not include any DARM algorithm code and Local knowledge is included as agent data component similarly global knowledge is included as agent data component in $Size_{GKDA\_P}$.

5.3.2 Cost Model involving n Distributed Sites

AL dispatches multiple cloned copy of each MA in parallel to n distributed sites. Mining is performed locally at each distributed site in parallel by *LKGA_P* and results are also transferred in parallel through *LKCA_P* to $S_{CENTRAL}$. Since mining is performed locally at each site simultaneously, the total time taken depends upon the time interval required by the site which takes longest time to perform data mining and return results. Therefore,

$$\begin{aligned} t_{darm} = &\max\left(t_{PDBFA}(S_{CENTRAL}, S_i) + t_{filtering}(S_i) + t_{PDBFA}(S_i, S_{CENTRAL})\right) + \\ &\max\left(t_{AAFFA}(S_{CENTRAL}, S_i) + t_{frequency}(S_i) + t_{AAFFA}(S_i, S_{CENTRAL})\right) + \\ &\max\left(t_{FMIDBGA}(S_{CENTRAL}, S_i) + t_{conversion}(S_i) + t_{FMIDBGA}(S_i, S_{CENTRAL})\right) + \\ &\max\left(t_{LKGA\_P}(S_{CENTRAL}, S_i) + t_{fim}(S_i) + t_{arm}(S_i) + t_{LKGA\_P}(S_i, S_{CENTRAL})\right) + \\ &\max\left(t_{LKCA\_P}(S_{CENTRAL}, S_i) + t_{LKCA\_P}(S_i, S_{CENTRAL})\right), \forall i = \{1 \ldots n\} \end{aligned} \tag{5.10}$$

Putting equation 5.10 in equation 5.1 and taking the longest time to dispatch the global knowledge to distributed sites, overall response time T for the DARM using AAeQARM-AAPDB framework is given as:

$$\begin{aligned} T = &\max\left(t_{PDBFA}(S_{CENTRAL}, S_i) + t_{filtering}(S_i) + t_{PDBFA}(S_i, S_{CENTRAL})\right) + \\ &\max\left(t_{AAFFA}(S_{CENTRAL}, S_i) + t_{frequency}(S_i) + t_{AAFFA}(S_i, S_{CENTRAL})\right) + \\ &\max\left(t_{FMIDBGA}(S_{CENTRAL}, S_i) + t_{conversion}(S_i) + t_{FMIDBGA}(S_i, S_{CENTRAL})\right) + \\ &\max\left(t_{LKGA\_P}(S_{CENTRAL}, S_i) + t_{fim}(S_i) + t_{arm}(S_i) + t_{LKGA\_P}(S_i, S_{CENTRAL})\right) + \\ &\max\left(t_{LKCA\_P}(S_{CENTRAL}, S_i) + t_{LKCA\_P}(S_i, S_{CENTRAL})\right) + t_{kigkg} + \\ &\max\left(t_{GKDA\_P}(S_{CENTRAL}, S_i) + t_{GKDA\_P}(S_i, S_{CENTRAL})\right) + t_{wait}, \forall i = \{1 \ldots n\} \end{aligned} \tag{5.11}$$

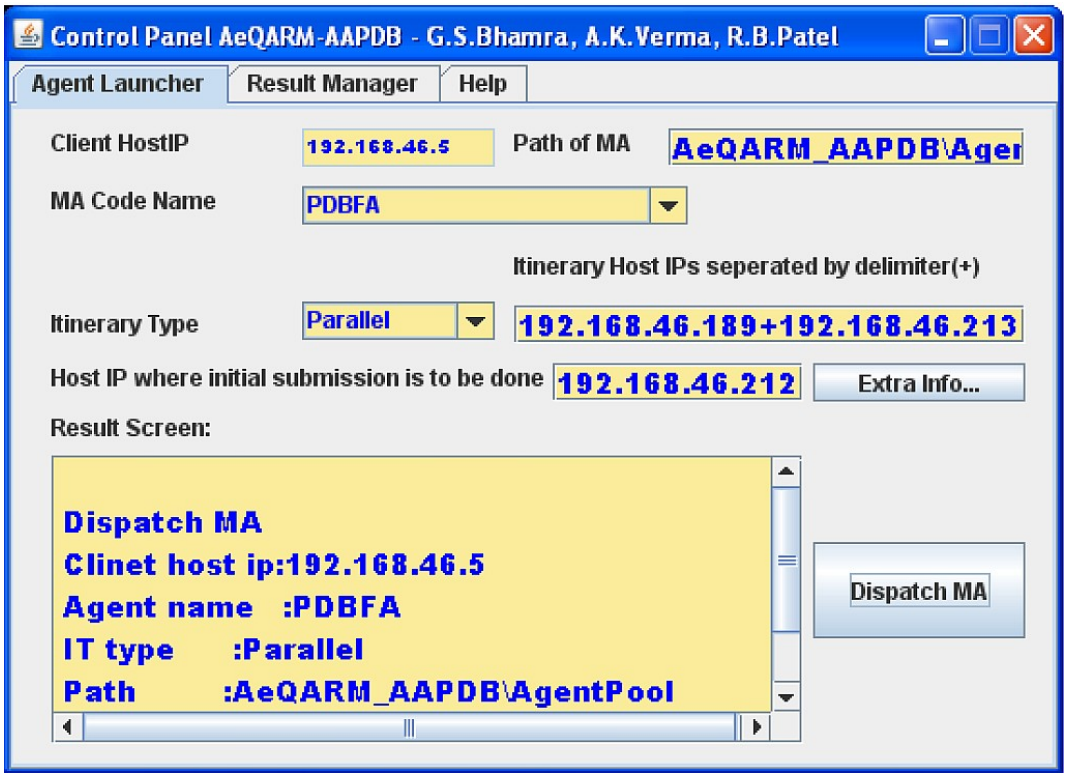

Figure 5.2: Control Panel of AeQARM-AAPDB

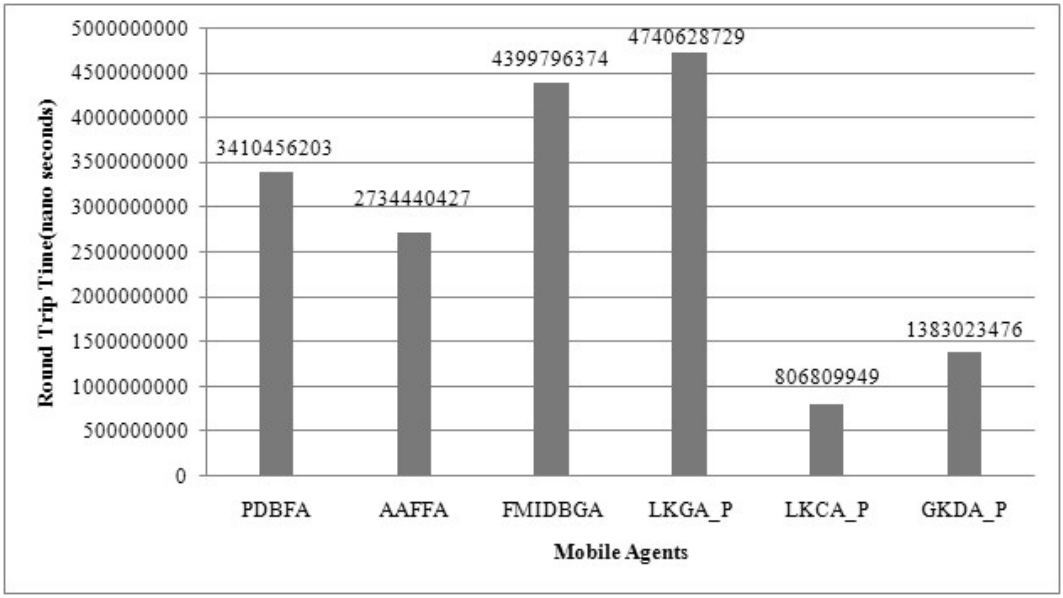

Figure 5.3: Round Trip Time taken by various MAs

5.4 Implementation and Performance Study

All the agents as well as Control panel of the system, as shown in Fig. 5.2, are implemented in Java. The required configuration for the deployment of the system is same as described

in Table 3.2 in Chapter 3 with additional deployment of *DM_AEE* at each distributed site and *AL* and *RM* at $S_{CENTRAL}$. Round Trip time taken by various MAs is shown in Fig. 5.3. CPU time consumed by various MAs at site S_1, S_2 and S_3 is shown in Fig. 5.4, Fig. 5.5 and Fig. 5.6, respectively. CPU time for *RIGKGA* is 102605790 nano seconds.

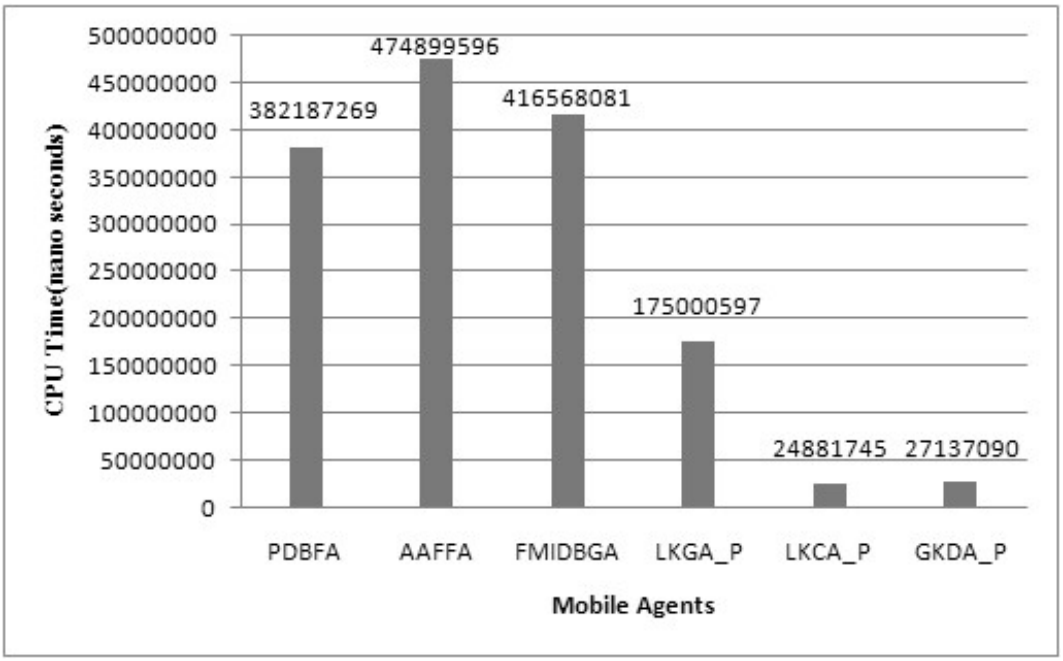

Figure 5.4: CPU Time taken by various MAs at site S_1

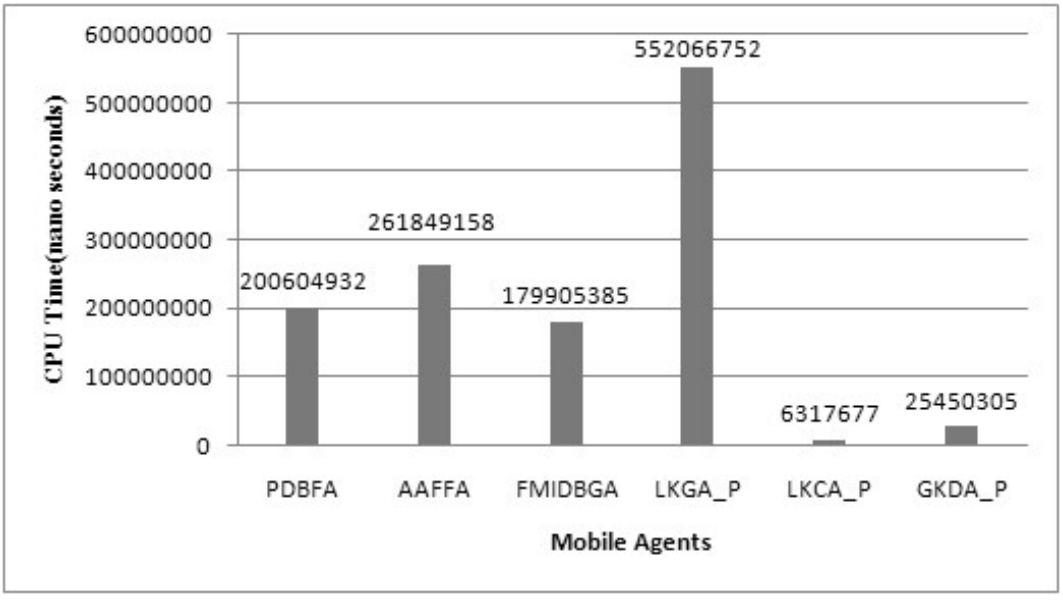

Figure 5.5: CPU Time taken by various MAs at site S_2

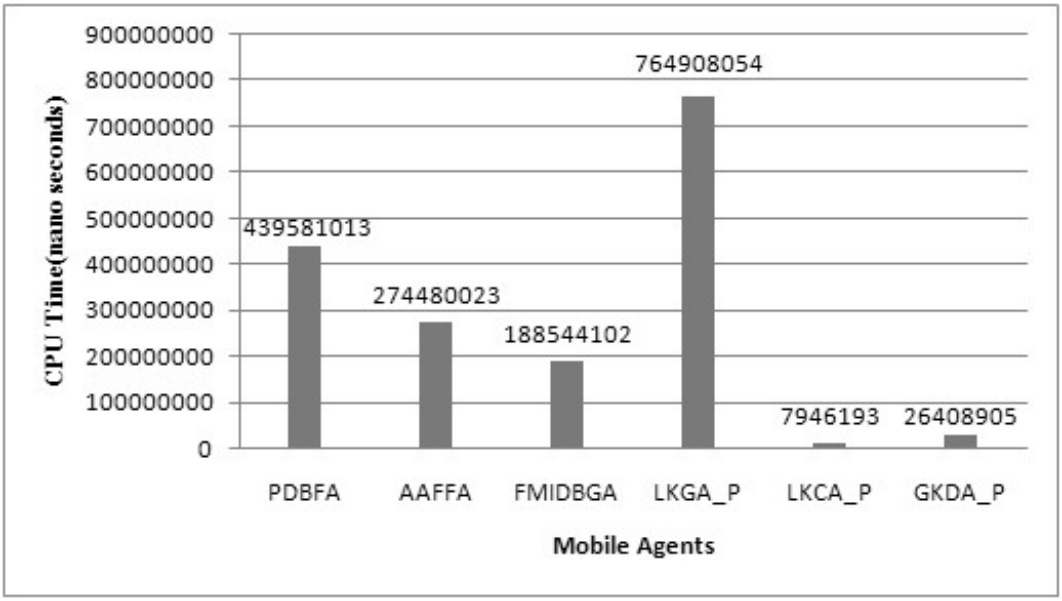

Figure 5.6: CPU Time taken by various MAs at site S_3

$L^{FI}_{k(i)}$ and $L^{FISC}_{k(i)}$ at distributed sites generated by *LKGA_P* agent with 20% *min_th_sup* are shown in Appendix D.1, D.2 and D.3. Locally strong association rules (L^{LSAR}_i) generated by *LKGA_P* for frequent item numbers at different sites are shown in Appendix D.4, D.6 and D.8 and the same for their corresponding amino acids frequency range are shown in Appendix D.5, D.7 and D.9. Globally frequent 2-itemsets (L^{GFI}_2) generated by *RIGKGA* is shown in Fig. 5.7 and globally strong association rules ($L^{GSAR}_{CENTRAL}$) for these globally frequent itemsets are shown in Fig. 5.8. When these item numbers are mapped with their corresponding amino acids frequency ranges then globally strong quantitative association rules for frequent 2 amino acids are obtained and shown in Fig. 5.9.

Sr. No.	2-Itemsets (L)
1	[16,92]
2	[16,151]
3	[16,152]
4	[16,271]
5	[16,287]
6	[91,271]
7	[92,271]
8	[151,271]
9	[152,271]
10	[197,271]
11	[271,287]

Figure 5.7: Globally frequent 2-itemsets

Sr. No.	Stong Association Rules for frequent 2-Itemsets		
	L	AR (support,confidence)	Site
1	[16, 92]	[92] => [16] (23%, 73%)	192.168.46.212
2		[92] => [16] (24%, 66%)	192.168.46.189
3		[92] => [16] (23%, 66%)	192.168.46.213
4	[16, 151]	[16] => [151] (44%, 59%)	192.168.46.212
5		[151] => [16] (44%, 82%)	
6		[151] => [16] (20%, 72%)	192.168.46.189
7		[151] => [16] (29%, 64%)	192.168.46.213
8	[16, 152]	[152] => [16] (24%, 72%)	192.168.46.212
9		[152] => [16] (23%, 62%)	192.168.46.189
10		[152] => [16] (25%, 68%)	192.168.46.213
11	[16, 271]	[16] => [271] (58%, 78%)	192.168.46.212
12		[271] => [16] (58%, 79%)	
13		[16] => [271] (38%, 64%)	192.168.46.189
14		[271] => [16] (38%, 66%)	
15		[16] => [271] (47%, 73%)	192.168.46.213
16		[271] => [16] (47%, 67%)	
17	[16, 287]	[287] => [16] (27%, 78%)	192.168.46.212
18		[287] => [16] (20%, 66%)	192.168.46.189
19		[287] => [16] (24%, 68%)	192.168.46.213
20	[91, 271]	[91] => [271] (43%, 82%)	192.168.46.212
21		[271] => [91] (43%, 59%)	
22		[91] => [271] (20%, 77%)	192.168.46.189
23		[91] => [271] (36%, 83%)	192.168.46.213
24		[271] => [91] (36%, 52%)	
25	[92, 271]	[92] => [271] (22%, 71%)	192.168.46.212
26		[92] => [271] (23%, 64%)	192.168.46.189
27		[92] => [271] (23%, 67%)	192.168.46.213
28	[151, 271]	[151] => [271] (43%, 80%)	192.168.46.212
29		[271] => [151] (43%, 59%)	
30		[151] => [271] (20%, 69%)	192.168.46.189
31		[151] => [271] (37%, 81%)	192.168.46.213
32		[271] => [151] (37%, 53%)	
33	[152, 271]	[152] => [271] (23%, 70%)	192.168.46.212
34		[152] => [271] (22%, 60%)	192.168.46.189
35		[152] => [271] (24%, 66%)	192.168.46.213
36	[197, 271]	[197] => [271] (29%, 80%)	192.168.46.212
37		[197] => [271] (20%, 73%)	192.168.46.189
38		[197] => [271] (26%, 76%)	192.168.46.213
39	[271, 287]	[287] => [271] (27%, 77%)	192.168.46.212
40		[287] => [271] (22%, 71%)	192.168.46.189
41		[287] => [271] (26%, 75%)	192.168.46.213

Figure 5.8: Globally strong association rules for frequent 2-itemsets

Sr. No.	Stong Quantitative Association Rules for 2-Itemsets(amino acids)		
	L	AR (support,confidence)	Site
1	{ <C:0..2> <H:3..5> }	{ <H:3..5> } => { <C:0..2> } (23%, 73%)	192.168.46.212
2		{ <H:3..5> } => { <C:0..2> } (24%, 66%)	192.168.46.189
3		{ <H:3..5> } => { <C:0..2> } (23%, 66%)	192.168.46.213
4	{ <C:0..2> <M:0..2> }	{ <C:0..2> } => { <M:0..2> } (44%, 59%)	192.168.46.212
5		{ <M:0..2> } => { <C:0..2> } (44%, 82%)	
6		{ <M:0..2> } => { <C:0..2> } (20%, 72%)	192.168.46.189
7		{ <M:0..2> } => { <C:0..2> } (29%, 64%)	192.168.46.213
8	{ <C:0..2> <M:3..5> }	{ <M:3..5> } => { <C:0..2> } (24%, 72%)	192.168.46.212
9		{ <M:3..5> } => { <C:0..2> } (23%, 62%)	192.168.46.189
10		{ <M:3..5> } => { <C:0..2> } (25%, 68%)	192.168.46.213
11	{ <C:0..2> <W:0..2> }	{ <C:0..2> } => { <W:0..2> } (58%, 78%)	192.168.46.212
12		{ <W:0..2> } => { <C:0..2> } (58%, 79%)	
13		{ <C:0..2> } => { <W:0..2> } (38%, 64%)	192.168.46.189
14		{ <W:0..2> } => { <C:0..2> } (38%, 66%)	
15		{ <C:0..2> } => { <W:0..2> } (47%, 73%)	192.168.46.213
16		{ <W:0..2> } => { <C:0..2> } (47%, 67%)	
17	{ <C:0..2> <Y:3..5> }	{ <Y:3..5> } => { <C:0..2> } (27%, 78%)	192.168.46.212
18		{ <Y:3..5> } => { <C:0..2> } (20%, 66%)	192.168.46.189
19		{ <Y:3..5> } => { <C:0..2> } (24%, 68%)	192.168.46.213
20	{ <H:0..2> <W:0..2> }	{ <H:0..2> } => { <W:0..2> } (43%, 82%)	192.168.46.212
21		{ <W:0..2> } => { <H:0..2> } (43%, 59%)	
22		{ <H:0..2> } => { <W:0..2> } (20%, 77%)	192.168.46.189
23		{ <H:0..2> } => { <W:0..2> } (36%, 83%)	192.168.46.213
24		{ <W:0..2> } => { <H:0..2> } (36%, 52%)	
25	{ <H:3..5> <W:0..2> }	{ <H:3..5> } => { <W:0..2> } (22%, 71%)	192.168.46.212
26		{ <H:3..5> } => { <W:0..2> } (23%, 64%)	192.168.46.189
27		{ <H:3..5> } => { <W:0..2> } (23%, 67%)	192.168.46.213
28	{ <M:0..2> <W:0..2> }	{ <M:0..2> } => { <W:0..2> } (43%, 80%)	192.168.46.212
29		{ <W:0..2> } => { <M:0..2> } (43%, 59%)	
30		{ <M:0..2> } => { <W:0..2> } (20%, 69%)	192.168.46.189
31		{ <M:0..2> } => { <W:0..2> } (37%, 81%)	192.168.46.213
32		{ <W:0..2> } => { <M:0..2> } (37%, 53%)	
33	{ <M:3..5> <W:0..2> }	{ <M:3..5> } => { <W:0..2> } (23%, 70%)	192.168.46.212
34		{ <M:3..5> } => { <W:0..2> } (22%, 60%)	192.168.46.189
35		{ <M:3..5> } => { <W:0..2> } (24%, 66%)	192.168.46.213
36	{ <Q:3..5> <W:0..2> }	{ <Q:3..5> } => { <W:0..2> } (29%, 80%)	192.168.46.212
37		{ <Q:3..5> } => { <W:0..2> } (20%, 73%)	192.168.46.189
38		{ <Q:3..5> } => { <W:0..2> } (26%, 76%)	192.168.46.213
39	{ <W:0..2> <Y:3..5> }	{ <Y:3..5> } => { <W:0..2> } (27%, 77%)	192.168.46.212
40		{ <Y:3..5> } => { <W:0..2> } (22%, 71%)	192.168.46.189
41		{ <Y:3..5> } => { <W:0..2> } (26%, 75%)	192.168.46.213

Figure 5.9: Globally strong association rules for frequent 2-amino acids

5.4.1 Discussion

The results for the globally strong amino acids, as shown in Fig. 5.9, reveals that

- The consequence part in all rules contains either Cysteine (C) or Tryptophan (W).
- The consequence part of these rules say that Cysteine (C) and Tryptophan (W) occur in frequency range 0-2.

- Cysteine(C) and Tryptophan(W) are the less frequent amino acids in proteins and in most proteins their frequency is close to zero.
- $< Cysteine(C) : 0..2 >$ is strongly associated with $< Histidine(H) : 3..5 >$, $< Methionine(M) : 0..2 >$, $< Methionine(M) : 3..5 >$, $< Tryptophan(W) : 0..2 >$, and $< Tyrosine(Y) : 3..5 >$.
- $< Tryptophan(W) : 0..2 >$ is strongly associated with $< Histidine(H) : 0..2 >$, $< Histidine(H) : 3..5 >$, $< Methionine(M) : 0..2 >$, $< Methionine(M) : 3..5 >$, $< Glutamine(Q) : 3..5 >$ and $< Tyrosine(Y) : 3..5 >$

Table 5.1: Comparison of proposed AeQARM-AAPDB with N. Gupta et al. [51] Model

Parameters	**N. Gupta et al. [51] Model**	**AeQARM-AAPDB**
ARM technique	It is based on the traditional central DW based ARM approach	It is based on the mobile agents based DARM approach
Data distribution	Distribution of the data is not taken into account	Data is distributed at sites and mined locally by mobile agents
Storage cost	Storage cost is increased as entire DW of protein records are stored at single site	Storage cost is reduced as protein dataset are stored on distributed sites
Computational cost	Computational cost is increased as entire ARM process is performed on single site	Computational cost is reduced as ARM process is performed locally at distributed sites by agents and only results are transferred at central site
Scalability	It is not a scalable system	It is scalable approach as adding more distributed sites would not affect the performance of the entire system
Cost Model for ARM task	No overall time modal is discussed for the ARM task	Overall time model is discussed for DARM task (equation 5.1). Various factors affecting the cost modal like size and density of the local dataset, number of other processes running at i^{th} site, the processor speed at i^{th} site and network bandwidth between two node are also discussed.
Computational time for ARM task	No computational time is measured for the ARM task.	System is implemented in Java and computational time for various agents involved for performing the DARM task is measured in nano seconds. Round trip time for mobile agents is also measured.

These results are similar to the findings in N. Gupta et al. [51] where protein sequences are taken from the SCOP Astral File v1.63 [11] for mining the global association rules in amino acids. Table 5.1 shows the qualitative comparison of the AeQARM-AAPDB system with N. Gupta et al. [51] Model.

The analysis shows that the proposed AeQARM-AAPDB framework has improved features and better performance than N. Gupta et al. [51] model.

5.5 Summary

DDM clubbed with MAS provides a rewarding solution in managing Big Data with ever increasing size. This chapter discusses a MAS called AeQARM-AAPDB to mine the strong quantitative association rule for among amino acids present in primary structure of the proteins from the distributed proteins data sets using intelligent agents. Such globally strong association rules are used in understanding of protein composition and are desirable for synthesis of artificial proteins. By applying multi-agent based distributed bio-data mining, the computing load can be balanced and the computational effort can be achieved in a parallel and scalable manner. To the best of our knowledge, this is the first systematic study to discover global associations between amino acids in a distributed environment using mobile agents.

Chapter 6

CONCLUSIONS

DM technique is used to extract interesting and useful patterns from large databases. ARs are used to discover the associations among items in a database. DARM task generates the globally strong association rules from the global frequent itemsets in a distributed environment.

The research described in this thesis has investigated the use of agent technology to provide the solution for DARM task especially in Bio-informatics domain. A scalable agent based framework for ARM of distributed data has been designed and implemented.

6.1 Contributions

This thesis may be considered as an approach that advocates the integration of DM and MAS especially in Bio-informatics. It also gives a picture of agent technology based DARM. Novelty factors for presented work includes the following:

1. **Tool for synthetic dataset generation:** A transactional dataset is required for mining the ARs from the frequent itemsets. This dataset can be a real dataset of any retail industry or can be a synthetic version generated by a tool. A software tool called Transactional Dataset Generator (TDSG) has been designed and implemented in Java language. It generates a Binary Dataset (BDS) as well as Transactional Dataset (TDS) corresponding to the Binary Dataset. *TDSG* takes three inputs - 1) total number of items in a transactional dataset; 2) total number of transactions; and 3) approximate density of the number of 1s in BDS. A 2-D array of integer values is created with number of columns equals number of items and number of rows equals number of transactions. Array elements can have either of the two binary values 0 or 1, where 1 represents the item in a transaction is purchased and 0 otherwise. Repeated attempts are made to get the desired density array with the help of Density Distributor component. If the BDS of required density is created then a TDS is generated. Each transaction in TDS consists of Transaction Identification (TID) and itemset, i.e., all the items purchased in that transaction. This tool generates two output text files, a BDS text file (e.g. BinaryDS20T10I.txt) for BDS and TDS text file (e.g. TDS20T10I.txt) for TDS and stores them at desired location in the local file system. It supports a maximum of 2^{32} number of items and

a maximum of 2^{32} number of transactions in a data set. Thus, it generates a Binary datasets of 2^{64} items. A TDS at i^{th} distributed site is termed as DB_i.

2. **Central DW based ARM:** Traditional central DW based approach for ARM is practically investigated with the help of a client-server based framework. The overall response time (T) for the ARM task performed using this approach is also formulated. Various components involved in this approach are:

   (a) ***Dataset Dispatcher (DD)***: It is a client application running at each distributed site S_i to dispatch DB_i over the network to the *Central Data Warehouse Manager (CDWM)*. *CDWM* is a server application running at $S_{CENTRAL}$. *DD* also keeps track of the time taken to dispatch DB_i from i^{th} site, i.e., $t_{dispatch}(S_i)$. Steps involved in the process is given in Algorithm 1 in Chapter 3.

   (b) ***Central Data Warehouse Manager (CDWM)***- is a multi-threaded server application running at $S_{CENTRAL}$. It handles all the incoming requests from *DD*. *CDWM* receives & stores each DB_i and waits continuously till all the clients have dispatched their DB_i. It also sends the data received time (t_{end}) back to the *DD*. Detailed steps are shown in Algorithm 2 in Chapter 3. Various other assistant components of *CDWM* application are:-

      i. ***Dataset Merger (DSM)*** merges all the collected set of DB_i from distributed sites into a single integrated unit ($DB = \bigcup_{i=1}^{n} DB_i$). Steps involved in the process are shown in Algorithm 3 in Chapter 3. The time taken in merging all the DB_i, i.e., $t_{merging}$ is also formulated.

      ii. ***Frequent Itemsets and Support Counts Generator (FISCG)***: It implements Apriori [5] algorithm to scan merged dataset DB with minimum threshold support (min_th_sup) and generates list of frequent itemsets (L^{FI}) and list of support counts for frequent itemsets (L^{FISC}). Detailed steps are shown in Algorithm 4 in Chapter 3. The time taken in frequent itemset mining of merged dataset (DB), i.e., t_{fim} is also formulated.

      iii. ***Strong Association Rule Harvester (SARH)***: It generates the list of strong association rules (L^{SAR}) from L^{FI} and L^{FISC} with the constraint of minimum threshold confidence (min_th_conf). Steps used by *SARH* are given in Algorithm 7 in Chapter 3. The time taken in association rule mining from L^{FI}, i.e., t_{arm} is also formulated.

The practical investigation and analysis indicates that this central DW based DM approach is ineffective because of storage, communication and computational costs involved in managing data. Scalability is one of the major issue where adding more sites would affect the performance of the system. Performance and scalability of a DM application can be increased by distributing workload among the sites which is possible when the DM is performed locally and only results are carried out at central site for mining global knowledge. The outcome of this approach suggested the use of agent technology for DARM task for the issue of scalability and global knowledge extraction.

3. **Agent Execution Environment for DARM:** An AEE is designed and implemented that acts as a distributed server application for managing a MAS for DARM task. It provides the appropriate functionality to MAs to execute, communicate (with other agents, users, and other platforms), migrate to other platform, manage itinerary and use system resources (local and global knowledge). This environment consists of the following three components:

   - ***Data Mining Agent Execution Environment (DM_AEE)***: It is the key component that acts as a Server. DM_AEE is deployed on any distributed sites (S_i) and is responsible for receiving, executing and migrating all the visiting DM agents. Steps are shown in Algorithm 8 in Chapter 4.
   - ***Agent Launcher (AL)***: It acts as a Client at $S_{CENTRAL}$ and launches the goal oriented DM agents on behalf of the user through a user interface to the DM_AEE running at the distributed sites. Agent Pool (or Zone) at $S_{CENTRAL}$ is a repository of all the Mobile as well as Stationary agents(SAs). In case of serial computing model, AL dispatches a specific single MA and it travels from node-to-node. In case of parallel computing model, it creates clones of the specific MA and dispatches each clone in parallel to all the distributed sites. AL also contacts the Result Manager (RM) for processing the $Briefcase$ of an agent. Detailed steps are given in Algorithm 9 in Chapter 4.
   - ***Result Manager (RM)***: It manages and processes the $Briefcase$ of all MAs. RM is either contacted by a MA for submitting its results or by AL for processing the results of specific MA. On completion of itinerary, each DM agent submits its results to RM which computes total round trip time ($TripTime$) of that MA and saves it in the $Briefcase$ of that agent. It is assumed that all the clones report their results to RM. Steps are defined in Algorithm 10 in Chapter 4.

4. **MAS for DARM:** A scalable MAS called Agent enriched Mining of Globally Strong Association Rules (AeMGSAR) for DARM task is designed and implemented using two computing models based on the itinerary of MAs. Performance of this system is compared with the traditional central DW based ARM approach.

   - **Serial computing model:** MAs in this model visit n distributed sites serially and performs their designated tasks. The overall response time for the DARM task performed using this model is also formulated. Serial itinerary used for MAs increases the overall cost of DARM task so a parallel computing model is designed.
   - **Parallel computing model:** Clones of MAs in this model visit n distributed sites in parallel and performs their designated tasks. On comparing with the serial computing model, it is found that overall response time for the DARM task involving n distributed sites is very less in case of parallel computing model of AeMGSAR.

   The comparative analysis on various parameters reveals that the proposed AeMGSAR framework has improved features and exhibit superior performance than the exist-

ing MADKDS [110], AFARMDD [59] and MADARM [87] frameworks for DARM task.

5. **A Case study in Bio-informatics:** As mining biological data is an emerging area at the intersection between bio-informatics and DDM, we have also taken the case of DARM in bio-informatics and designed a MAS called Agent enriched Quantitative Association Rules Mining for Amino Acids in distributed Protein Data Banks (AeQARM-AAPDB) for mining the quantitative ARs for amino acids in proteins. Comparison of the AeQARM-AAPDB system with N. Gupta et al. [51] approach on various parameters shows that the proposed AeQARM-AAPDB framework has improved features and better performance than N. Gupta et al. [51] model.

6.2 Future Scope

The challenges and issues of DARM task clearly indicated the need for creation of intelligent agent based scalable DM systems. A foundation has been established for both DM research and genuine application based DARM. There are many other directions in which the work can be taken forward. A few of them are summarized here.

1. **Agent Security:** Runtine environment for the framework relies on the in-built security system of Java language for the incoming MAs at each site. Security can be provided to MAs by a central agency so that authenticity of each visiting MA can be verified and only trusted agents are allow to access the resources of available at the distributed site.

2. **Privacy Preserving:** Distributed data may be highly sensitive and private. Privacy concerns may prevent the parties from directly sharing the data, and some types of information about the data. Cryptography techniques can be applied on the shared knowledge among sites to protect the privacy.

3. **Fast algorithms:** Present framework is tested using Apriori algorithm [5] for mining the frequent itemsets. Other fast algorithms can be designed to reduce the computational cost.

4. **Mining of other biological databases:** Mining distributed biological data is an emergent area of research as size of such datasets are increasing exponentially. Present work study the associations among amino acids in proteins. Other biological datasets like Genome can be considered for finding associations among the constituent elements.

5. **Fault Tolerance:** This MAS is not fault tolerant. Site failure and recovery may be taken into account. The permission for non-reporting clones of MA may also be incorporated.

6. **IPv6 support for addressing:** Currently the underlying AEE supports IPv4 network addressing scheme. It may be modified to support IPv6.

Appendix A - Synthetic Datasets

A.1 *BDS3500T10I.txt and corresponding TDS3500T10I.txt(DB_1) at site S_1*

These synthetic binary and transactional datasets of 3500 records are created by *TDSG* tool. In the binary version each column head represents the item number and each row represents a transaction where integer '1' is used for a purchased item and '0' is used if it is not purchased. The corresponding transactional version has a Transaction ID (TID) for each transaction and Itemset is the the set of all the purchased items for that transaction.

1	2	3	4	5	6	7	8	9	10
1	1	0	0	0	1	1	0	1	1
1	0	1	0	1	1	1	0	1	1
1	0	1	0	0	1	1	1	1	0
1	1	1	1	0	1	1	0	1	0
1	1	1	1	0	1	0	1	1	0
1	0	0	1	1	0	0	0	1	1
1	1	1	0	0	1	1	1	1	1
1	0	1	1	0	1	1	0	1	0
1	1	0	0	0	1	0	0	1	1
1	1	0	1	1	1	0	1	1	0
1	0	1	1	0	1	0	0	1	1
1	1	0	1	0	1	1	1	1	1
1	0	0	1	1	1	1	0	1	1
1	1	1	1	1	1	0	0	1	0
1	1	1	0	0	1	0	1	1	0
1	1	1	1	0	1	1	0	1	0
1	1	1	0	0	1	1	0	1	0
1	0	0	1	1	1	1	0	1	1
1	1	0	0	0	1	0	1	1	0
1	1	0	1	0	1	1	1	1	1
⋮									

TID	Itemset
T1	1 2 6 7 9 10
T2	1 3 5 6 7 9 10
T3	1 3 6 7 8 9
T4	1 2 3 4 6 7 9
T5	1 2 3 4 6 8 9
T6	1 4 5 9 10
T7	1 2 3 6 7 8 9 10
T8	1 3 4 6 7 9
T9	1 2 6 9 10
T10	1 2 4 5 6 8 9
T11	1 3 4 6 9 10
T12	1 2 4 6 7 8 9 10
T13	1 4 5 6 7 9 10
T14	1 2 3 4 5 6 9
T15	1 2 3 6 8 9
T16	1 2 3 4 6 7 9
T17	1 2 3 6 7 9
T18	1 4 5 6 7 9 10
T19	1 2 6 8 9
T20	1 2 4 6 7 8 9 10
⋮	

A.2 *BDS3850T10I.txt and corresponding TDS3850T10I.txt(DB_2) at site S_2*

These synthetic binary and transactional datasets of 3850 records are created by TDSG tool. In the binary version each column head represents the item number and each row represents a transaction where integer '1' is used for a purchased item and '0' is used if it is not purchased. The corresponding transactional version has a Transaction ID (TID) for each transaction and Itemset is the the set of all the purchased items for that transaction.

1	2	3	4	5	6	7	8	9	10
1	1	0	1	0	0	1	1	1	0
1	0	0	1	0	1	1	1	1	0
1	1	1	1	0	0	1	1	0	0
1	1	0	0	0	0	1	1	1	0
1	1	0	1	0	0	1	1	0	0
1	0	1	1	0	0	1	1	1	0
0	0	1	0	0	1	1	1	1	0
1	0	1	1	0	0	1	1	1	0
1	0	0	1	1	0	1	1	0	0
1	0	0	1	0	0	1	1	1	0
1	0	0	0	0	0	0	1	0	1
1	0	1	0	1	0	1	1	0	0
1	0	1	1	0	0	1	1	0	0
1	0	1	0	0	0	0	1	1	0
1	0	0	0	0	0	1	1	1	0
1	1	1	1	0	0	1	1	1	1
1	0	0	1	0	0	1	1	1	0
1	0	0	0	0	0	1	1	0	0
1	1	1	1	0	0	1	1	1	0
1	0	0	1	0	0	1	1	0	0
⋮									

TID	Itemset
T1	1 2 4 7 8 9
T2	1 4 6 7 8 9
T3	1 2 3 4 7 8
T4	1 2 7 8 9
T5	1 2 4 7 8
T6	1 3 4 7 8 9
T7	3 6 7 8 9
T8	1 3 4 7 8 9
T9	1 4 5 7 8
T10	1 4 7 8 9
T11	1 8 10
T12	1 3 5 7 8
T13	1 3 4 7 8
T14	1 3 8 9
T15	1 7 8 9
T16	1 2 3 4 7 8 9 10
T17	1 4 7 8 9
T18	1 7 8
T19	1 2 3 4 7 8 9
T20	1 4 7 8
⋮	

A.3 *BDS3900T10I.txt and corresponding TDS3900T10I.txt(DB_3) at site S_3*

These synthetic binary and transactional datasets of 3900 records are created by TDSG tool. In the binary version each column head represents the item number and each row represents a transaction where integer '1' is used for a purchased item and '0' is used if it is not purchased. The corresponding transactional version has a Transaction ID (TID) for each transaction and Itemset is the the set of all the purchased items for that transaction.

1	2	3	4	5	6	7	8	9	10
1	1	0	1	1	1	1	1	1	1
0	1	1	0	0	1	0	1	1	1
1	1	1	1	1	1	1	1	1	1
0	0	1	1	0	1	0	1	1	1
1	1	1	1	0	1	1	1	1	1
1	0	0	0	0	1	1	1	0	1
1	0	1	1	0	0	0	1	1	1
0	1	0	1	1	1	0	1	1	0
1	1	1	1	0	0	1	1	1	1
0	1	1	1	0	0	0	1	1	1
0	1	0	1	0	0	0	1	1	1
1	1	1	1	0	0	0	1	0	1
0	0	0	1	0	0	0	1	0	1
1	1	1	1	0	0	0	1	1	1
0	0	1	0	0	1	1	1	1	1
1	0	0	1	0	1	1	1	1	1
0	1	0	1	1	1	1	1	1	1
1	1	1	0	1	0	1	1	0	1
0	0	1	1	0	1	0	1	1	1
1	1	0	1	1	0	1	1	0	1
⋮									

TID	Itemset
T1	1 2 4 5 6 7 8 9 10
T2	2 3 6 8 9 10
T3	1 2 3 4 5 6 7 8 9 10
T4	3 4 6 8 9 10
T5	1 2 3 4 6 7 8 9 10
T6	1 6 7 8 10
T7	1 3 4 8 9 10
T8	2 4 5 6 8 9
T9	1 2 3 4 7 8 9 10
T10	2 3 4 8 9 10
T11	2 4 8 9 10
T12	1 2 3 4 8 10
T13	4 8 10
T14	1 2 3 4 8 9 10
T15	3 6 7 8 9 10
T16	1 4 6 7 8 9 10
T17	2 4 5 6 7 8 9 10
T18	1 2 3 5 7 8 10
T19	3 4 6 8 9 10
T20	1 2 4 5 7 8 10
⋮	

Appendix B - Resultant Knowledge of AeMGSAR System

B.1 $L^{FI}_{k(1)}$ *and* $L^{FISC}_{k(1)}$ *at site* S_1

List of frequent k-itemset, i.e., $L^{FI}_{k(1)}$ is represented by column **L** and column **SC** shows the support count of corresponding frequent k-itemset ($L^{FISC}_{k(1)}$) at site S_1. These frequent itemsets and their support counts are obtained by processing the synthetic dataset (DB_1) as shown in Appendix A.1.

S.n.	1-Itemsets		2-Itemsets		3-Itemsets		4-Itemsets		5-Itemsets		6-Itemsets	
	L	SC	L	SC	L	SC	L	SC	L	SC	L	SC
1	[1]	3444	[1, 2]	2054	[1, 2, 3]	829	[1, 2, 3, 6]	781	[1, 2, 3, 6, 9]	718	[1, 2, 4, 6, 7, 9]	791
2	[2]	2087	[1, 3]	1377	[1, 2, 4]	1306	[1, 2, 3, 9]	760	[1, 2, 4, 6, 7]	870		
3	[3]	1403	[1, 4]	2191	[1, 2, 6]	1942	[1, 2, 4, 6]	1231	[1, 2, 4, 6, 9]	1134		
4	[4]	2228	[1, 5]	810	[1, 2, 7]	1452	[1, 2, 4, 7]	917	[1, 2, 4, 7, 9]	833		
5	[5]	826	[1, 6]	3260	[1, 2, 8]	725	[1, 2, 4, 9]	1203	[1, 2, 6, 7, 9]	1248		
6	[6]	3316	[1, 7]	2443	[1, 2, 9]	1887	[1, 2, 6, 7]	1375	[1, 2, 6, 9, 10]	744		
7	[7]	2482	[1, 8]	1183	[1, 2, 10]	865	[1, 2, 6, 9]	1785	[1, 3, 4, 6, 9]	753		
8	[8]	1203	[1, 9]	3158	[1, 3, 4]	882	[1, 2, 6, 10]	810	[1, 3, 6, 7, 9]	828		
9	[9]	3210	[1, 10]	1466	[1, 3, 6]	1299	[1, 2, 7, 9]	1317	[1, 4, 6, 7, 9]	1346		
10	[10]	1490	[2, 3]	844	[1, 3, 7]	974	[1, 2, 9, 10]	796	[1, 4, 6, 9, 10]	822		
11			[2, 4]	1327	[1, 3, 9]	1260	[1, 3, 4, 6]	828	[1, 6, 7, 8, 9]	721		
12			[2, 6]	1975	[1, 4, 6]	2065	[1, 3, 4, 9]	802	[1, 6, 7, 9, 10]	881		
13			[2, 7]	1475	[1, 4, 7]	1561	[1, 3, 6, 7]	916	[2, 4, 6, 7, 9]	807		
14			[2, 8]	738	[1, 4, 8]	765	[1, 3, 6, 9]	1190				
15			[2, 9]	1919	[1, 4, 9]	2014	[1, 3, 7, 9]	880				
16			[2, 10]	878	[1, 4, 10]	962	[1, 4, 6, 7]	1479				
17			[3, 4]	898	[1, 5, 6]	751	[1, 4, 6, 8]	725				
18			[3, 6]	1325	[1, 5, 9]	755	[1, 4, 6, 9]	1895				
19			[3, 7]	992	[1, 6, 7]	2314	[1, 4, 6, 10]	906				
20			[3, 9]	1282	[1, 6, 8]	1124	[1, 4, 7, 9]	1422				
21			[4, 6]	2102	[1, 6, 9]	2990	[1, 4, 9, 10]	876				
22			[4, 7]	1588	[1, 6, 10]	1377	[1, 6, 7, 8]	799				
23			[4, 8]	777	[1, 7, 8]	845	[1, 6, 7, 9]	2104				
24			[4, 9]	2049	[1, 7, 9]	2220	[1, 6, 7, 10]	969				
25			[4, 10]	976	[1, 7, 10]	1036	[1, 6, 8, 9]	1019				

B.2 $L^{FI}_{k(2)}$ *and* $L^{FISC}_{k(2)}$ *at site* S_2

List of frequent k-itemset, i.e., $L^{FI}_{k(2)}$ is represented by column **L** and column **SC** shows the support count of corresponding frequent k-itemset ($L^{FISC}_{k(2)}$) at site S_2. These frequent itemsets and their support counts are obtained by processing the synthetic dataset (DB_2) as shown in Appendix A.2.

S.n.	1-Itemsets		2-Itemsets		3-Itemsets		4-Itemsets	
	L	SC	L	SC	L	SC	L	SC
1	[1]	2980	[1, 3]	1365	[1, 3, 7]	906	[1, 3, 7, 8]	880
2	[3]	1771	[1, 4]	1571	[1, 3, 8]	1330	[1, 3, 8, 9]	799
3	[4]	2054	[1, 7]	1980	[1, 3, 9]	827	[1, 4, 7, 8]	1036
4	[6]	859	[1, 8]	2890	[1, 4, 7]	1060	[1, 4, 8, 9]	948
5	[7]	2579	[1, 9]	1836	[1, 4, 8]	1533	[1, 7, 8, 9]	1190
6	[8]	3745	[3, 4]	944	[1, 4, 9]	977	[4, 7, 8, 9]	870
7	[9]	2399	[3, 7]	1182	[1, 7, 8]	1918		
8	[10]	1007	[3, 8]	1729	[1, 7, 9]	1234		
9			[3, 9]	1089	[1, 8, 9]	1772		
10			[4, 7]	1388	[3, 4, 8]	927		
11			[4, 8]	2009	[3, 7, 8]	1149		
12			[4, 9]	1292	[3, 8, 9]	1058		
13			[6, 8]	835	[4, 7, 8]	1357		
14			[7, 8]	2503	[4, 7, 9]	893		
15			[7, 9]	1634	[4, 8, 9]	1260		
16			[8, 9]	2325	[7, 8, 9]	1581		
17			[8, 10]	976				

B.3 $L^{FI}_{k(3)}$ *and* $L^{FISC}_{k(3)}$ *at site* S_3

List of frequent k-itemset, i.e., $L^{FI}_{k(3)}$ is represented by column **L** and column **SC** shows the support count of corresponding frequent k-itemset ($L^{FISC}_{k(3)}$) at site S_3. These frequent itemsets and their support counts are obtained by processing the synthetic dataset (DB_3) as shown in Appendix A.3.

S.n.	1-Itemsets		2-Itemsets		3-Itemsets		4-Itemsets		5-Itemsets	
	L	SC	L	SC	L	SC	L	SC	L	SC
1	[1]	2087	[1, 2]	1486	[1, 2, 3]	873	[1, 2, 3, 8]	861	[2, 3, 4, 6, 8]	787
2	[2]	2816	[1, 3]	1204	[1, 2, 4]	1155	[1, 2, 4, 8]	1138	[2, 3, 4, 8, 10]	828
3	[3]	2213	[1, 4]	1594	[1, 2, 6]	920	[1, 2, 6, 8]	909	[2, 4, 6, 8, 9]	799
4	[4]	2995	[1, 5]	790	[1, 2, 8]	1465	[1, 2, 8, 9]	894	[2, 4, 6, 8, 10]	880
5	[5]	1457	[1, 6]	1295	[1, 2, 9]	910	[1, 2, 8, 10]	977	[2, 4, 8, 9, 10]	844
6	[6]	2431	[1, 7]	1028	[1, 2, 10]	989	[1, 3, 4, 8]	925		
7	[7]	1912	[1, 8]	2055	[1, 3, 4]	937	[1, 4, 6, 8]	982		
8	[8]	3833	[1, 9]	1283	[1, 3, 8]	1188	[1, 4, 8, 9]	962		
9	[9]	2384	[1, 10]	1380	[1, 3, 10]	787	[1, 4, 8, 10]	1041		
10	[10]	2590	[2, 3]	1622	[1, 4, 6]	994	[1, 6, 8, 9]	802		
11			[2, 4]	2175	[1, 4, 7]	783	[1, 6, 8, 10]	837		
12			[2, 5]	1047	[1, 4, 8]	1572	[1, 8, 9, 10]	832		
13			[2, 6]	1738	[1, 4, 9]	978	[2, 3, 4, 6]	797		
14			[2, 7]	1370	[1, 4, 10]	1055	[2, 3, 4, 8]	1256		
15			[2, 8]	2770	[1, 6, 8]	1278	[2, 3, 4, 9]	790		
16			[2, 9]	1723	[1, 6, 9]	811	[2, 3, 4, 10]	838		
17			[2, 10]	1870	[1, 6, 10]	849	[2, 3, 6, 8]	1004		
18			[3, 4]	1719	[1, 7, 8]	1010	[2, 3, 8, 9]	991		
19			[3, 5]	822	[1, 8, 9]	1264	[2, 3, 8, 10]	1048		
20			[3, 6]	1375	[1, 8, 10]	1360	[2, 4, 5, 8]	804		
21			[3, 7]	1087	[1, 9, 10]	842	[2, 4, 6, 8]	1327		
22			[3, 8]	2184	[2, 3, 4]	1272	[2, 4, 6, 9]	810		
23			[3, 9]	1360	[2, 3, 6]	1018	[2, 4, 6, 10]	892		
24			[3, 10]	1443	[2, 3, 8]	1600	[2, 4, 7, 8]	1039		
25			[4, 5]	1128	[2, 3, 9]	1006	[2, 4, 8, 9]	1299		

B.4 L_1^{LSAR} *at site* S_1

Column **L** represents frequent k-itemset and column **AR(support,confidence)** shows the list of locally strong association rules, i.e., L_1^{LSAR} at site S_1. Each strong rule has its associated support and confidence factor. The minimum threshold support is taken as 20% and minimum threshold confidence as 50% for generating the strong rules by making use of the data as shown in Appendix B.1.

	Strong ARs for Frequent 6-itemsets		Strong ARs for Frequent 5-itemsets	
Sr. No.	L	AR(support,confidence)	L	AR(support,confidence)
1	[1, 2, 4, 6, 7, 9]	[2, 4] => [1, 6, 7, 9] (22%, 59%)	[1, 2, 3, 6, 9]	[3] => [1, 2, 6, 9] (20%, 51%)
2		[2, 7] => [1, 4, 6, 9] (22%, 53%)		[1, 3] => [2, 6, 9] (20%, 52%)
3		[1, 2, 4] => [6, 7, 9] (22%, 60%)		[2, 3] => [1, 6, 9] (20%, 85%)
4		[1, 2, 7] => [4, 6, 9] (22%, 54%)		[3, 6] => [1, 2, 9] (20%, 54%)
5		[1, 4, 7] => [2, 6, 9] (22%, 50%)		[3, 9] => [1, 2, 6] (20%, 56%)
6		[2, 4, 6] => [1, 7, 9] (22%, 63%)		[1, 2, 3] => [6, 9] (20%, 86%)
7		[2, 4, 7] => [1, 6, 9] (22%, 84%)		[1, 3, 6] => [2, 9] (20%, 55%)
8		[2, 4, 9] => [1, 6, 7] (22%, 64%)		[1, 3, 9] => [2, 6] (20%, 56%)
9		[2, 6, 7] => [1, 4, 9] (22%, 56%)		[2, 3, 6] => [1, 9] (20%, 90%)
10		[2, 7, 9] => [1, 4, 6] (22%, 59%)		[2, 3, 9] => [1, 6] (20%, 92%)
11		[4, 6, 7] => [1, 2, 9] (22%, 52%)		[3, 6, 9] => [1, 2] (20%, 59%)
12		[4, 7, 9] => [1, 2, 6] (22%, 54%)		[1, 2, 3, 6] => [9] (20%, 91%)
13		[1, 2, 4, 6] => [7, 9] (22%, 64%)		[1, 2, 3, 9] => [6] (20%, 94%)
14		[1, 2, 4, 7] => [6, 9] (22%, 86%)		[1, 3, 6, 9] => [2] (20%, 60%)
15		[1, 2, 4, 9] => [6, 7] (22%, 65%)		[2, 3, 6, 9] => [1] (20%, 98%)
16		[1, 2, 6, 7] => [4, 9] (22%, 57%)	[1, 2, 4, 6, 7]	[2, 4] => [1, 6, 7] (24%, 65%)
17		[1, 2, 7, 9] => [4, 6] (22%, 60%)		[2, 7] => [1, 4, 6] (24%, 58%)
18		[1, 4, 6, 7] => [2, 9] (22%, 53%)		[4, 7] => [1, 2, 6] (24%, 54%)
19		[1, 4, 7, 9] => [2, 6] (22%, 55%)		[1, 2, 4] => [6, 7] (24%, 66%)
20		[2, 4, 6, 7] => [1, 9] (22%, 89%)		[1, 2, 7] => [4, 6] (24%, 59%)
21		[2, 4, 6, 9] => [1, 7] (22%, 68%)		[1, 4, 7] => [2, 6] (24%, 55%)
22		[2, 4, 7, 9] => [1, 6] (22%, 93%)		[2, 4, 6] => [1, 7] (24%, 69%)
23		[2, 6, 7, 9] => [1, 4] (22%, 62%)		[2, 4, 7] => [1, 6] (24%, 93%)
24		[4, 6, 7, 9] => [1, 2] (22%, 57%)		[2, 6, 7] => [1, 4] (24%, 62%)
25		[1, 2, 4, 6, 7] => [9] (22%, 90%)		[4, 6, 7] => [1, 2] (24%, 57%)
26		[1, 2, 4, 6, 9] => [7] (22%, 69%)		[1, 2, 4, 6] => [7] (24%, 70%)
27		[1, 2, 4, 7, 9] => [6] (22%, 94%)		[1, 2, 4, 7] => [6] (24%, 94%)
28		[1, 2, 6, 7, 9] => [4] (22%, 63%)		[1, 2, 6, 7] => [4] (24%, 63%)
29		[1, 4, 6, 7, 9] => [2] (22%, 58%)		[1, 4, 6, 7] => [2] (24%, 58%)
30		[2, 4, 6, 7, 9] => [1] (22%, 98%)		[2, 4, 6, 7] => [1] (24%, 98%)
31			[1, 2, 4, 6, 9]	[2] => [1, 4, 6, 9] (32%, 54%)
32				[4] => [1, 2, 6, 9] (32%, 50%)
33				[1, 2] => [4, 6, 9] (32%, 55%)
34				[1, 4] => [2, 6, 9] (32%, 51%)
35				[2, 4] => [1, 6, 9] (32%, 85%)
36				[2, 6] => [1, 4, 9] (32%, 57%)
37				[2, 9] => [1, 4, 6] (32%, 59%)
38				[4, 6] => [1, 2, 9] (32%, 53%)
39				[4, 9] => [1, 2, 6] (32%, 55%)
40				[1, 2, 4] => [6, 9] (32%, 86%)
41				[1, 2, 6] => [4, 9] (32%, 58%)
42				[1, 2, 9] => [4, 6] (32%, 60%)
43				[1, 4, 6] => [2, 9] (32%, 54%)
44				[1, 4, 9] => [2, 6] (32%, 56%)

B.5 L_2^{LSAR} *at site* S_2

Column **L** represents frequent k-itemset and column **AR(support,confidence)** shows the list of locally strong association rules, i.e., L_2^{LSAR} at site S_2. Each strong rule has its associated support and confidence factor. The minimum threshold support is taken as 20% and minimum threshold confidence as 50% for generating the strong rules by making use of the data as shown in Appendix B.2.

	Strong ARs for Frequent 4-itemsets		Strong ARs for Frequent 3-itemsets	
Sr. No.	L	AR(support,confidence)	L	AR(support,confidence)
1	[1, 3, 7, 8]	[1, 3] => [7, 8] (22%, 64%)	[1, 3, 7]	[3] => [1, 7] (23%, 51%)
2		[3, 7] => [1, 8] (22%, 74%)		[1, 3] => [7] (23%, 66%)
3		[3, 8] => [1, 7] (22%, 50%)		[3, 7] => [1] (23%, 76%)
4		[1, 3, 7] => [8] (22%, 97%)	[1, 3, 8]	[3] => [1, 8] (34%, 75%)
5		[1, 3, 8] => [7] (22%, 66%)		[1, 3] => [8] (34%, 97%)
6		[3, 7, 8] => [1] (22%, 76%)		[3, 8] => [1] (34%, 76%)
7	[1, 3, 8, 9]	[1, 3] => [8, 9] (20%, 58%)	[1, 3, 9]	[1, 3] => [9] (21%, 60%)
8		[3, 9] => [1, 8] (20%, 73%)		[3, 9] => [1] (21%, 75%)
9		[1, 3, 8] => [9] (20%, 60%)	[1, 4, 7]	[4] => [1, 7] (27%, 51%)
10		[1, 3, 9] => [8] (20%, 96%)		[1, 4] => [7] (27%, 67%)
11		[3, 8, 9] => [1] (20%, 75%)		[1, 7] => [4] (27%, 53%)
12	[1, 4, 7, 8]	[4] => [1, 7, 8] (26%, 50%)		[4, 7] => [1] (27%, 76%)
13		[1, 4] => [7, 8] (26%, 65%)	[1, 4, 8]	[1] => [4, 8] (39%, 51%)
14		[1, 7] => [4, 8] (26%, 52%)		[4] => [1, 8] (39%, 74%)
15		[4, 7] => [1, 8] (26%, 74%)		[1, 4] => [8] (39%, 97%)
16		[4, 8] => [1, 7] (26%, 51%)		[1, 8] => [4] (39%, 53%)
17		[1, 4, 7] => [8] (26%, 97%)		[4, 8] => [1] (39%, 76%)
18		[1, 4, 8] => [7] (26%, 67%)	[1, 4, 9]	[1, 4] => [9] (25%, 62%)
19		[1, 7, 8] => [4] (26%, 54%)		[1, 9] => [4] (25%, 53%)
20		[4, 7, 8] => [1] (26%, 76%)		[4, 9] => [1] (25%, 75%) ▪▪▪
21	[1, 4, 8, 9]	[1, 4] => [8, 9] (24%, 60%)	[1, 7, 8]	[1] => [7, 8] (49%, 64%)
22		[1, 9] => [4, 8] (24%, 51%)		[7] => [1, 8] (49%, 74%)
23		[4, 9] => [1, 8] (24%, 73%)		[8] => [1, 7] (49%, 51%)
24		[1, 4, 8] => [9] (24%, 61%)		[1, 7] => [8] (49%, 96%)
25		[1, 4, 9] => [8] (24%, 97%)		[1, 8] => [7] (49%, 66%)
26		[1, 8, 9] => [4] (24%, 53%)		[7, 8] => [1] (49%, 76%)
27		[4, 8, 9] => [1] (24%, 75%)	[1, 7, 9]	[9] => [1, 7] (32%, 51%)
28	[1, 7, 8, 9]	[1, 7] => [8, 9] (30%, 60%)		[1, 7] => [9] (32%, 62%)
29		[1, 9] => [7, 8] (30%, 64%)		[1, 9] => [7] (32%, 67%)
30		[7, 9] => [1, 8] (30%, 72%)		[7, 9] => [1] (32%, 75%)
31		[8, 9] => [1, 7] (30%, 51%)	[1, 8, 9]	[1] => [8, 9] (46%, 59%)
32		[1, 7, 8] => [9] (30%, 62%)		[9] => [1, 8] (46%, 73%)
33		[1, 7, 9] => [8] (30%, 96%)		[1, 8] => [9] (46%, 61%)
34		[1, 8, 9] => [7] (30%, 67%)		[1, 9] => [8] (46%, 96%)
35		[7, 8, 9] => [1] (30%, 75%)		[8, 9] => [1] (46%, 76%)
36	[4, 7, 8, 9]	[4, 7] => [8, 9] (22%, 62%)	[3, 4, 8]	[3] => [4, 8] (24%, 52%)
37		[4, 9] => [7, 8] (22%, 67%)		[3, 4] => [8] (24%, 98%)
38		[7, 9] => [4, 8] (22%, 53%)		[3, 8] => [4] (24%, 53%)
39		[4, 7, 8] => [9] (22%, 64%)	[3, 7, 8]	[3] => [7, 8] (29%, 64%)
40		[4, 7, 9] => [8] (22%, 97%)		[3, 7] => [8] (29%, 97%)
41		[4, 8, 9] => [7] (22%, 69%)		[3, 8] => [7] (29%, 66%)
42		[7, 8, 9] => [4] (22%, 55%)	[3, 8, 9]	[3] => [8, 9] (27%, 59%)

⋮

B.6 L_3^{LSAR} *at site* S_3

Column **L** represents frequent k-itemset and column **AR(support,confidence)** shows the list of locally strong association rules, i.e., L_3^{LSAR} at site S_3. Each strong rule has its associated support and confidence factor. The minimum threshold support is taken as 20% and minimum threshold confidence as 50% for generating the strong rules by making use of the data as shown in Appendix B.3.

	Strong ARs for Frequent 5-itemsets		Strong ARs for Frequent 4-itemsets	
Sr. No.	L	AR(support,confidence)	L	AR(support,confidence)
1	[2, 3, 4, 6, 8]	[3, 6] => [2, 4, 8] (20%, 57%)	[1, 2, 3, 8]	[1, 2] => [3, 8] (22%, 57%)
2		[2, 3, 4] => [6, 8] (20%, 61%)		[1, 3] => [2, 8] (22%, 71%)
3		[2, 3, 6] => [4, 8] (20%, 77%)		[2, 3] => [1, 8] (22%, 53%)
4		[2, 4, 6] => [3, 8] (20%, 58%)		[1, 2, 3] => [8] (22%, 98%)
5		[3, 4, 6] => [2, 8] (20%, 74%)		[1, 2, 8] => [3] (22%, 58%)
6		[3, 6, 8] => [2, 4] (20%, 57%)		[1, 3, 8] => [2] (22%, 72%)
7		[2, 3, 4, 6] => [8] (20%, 98%)		[2, 3, 8] => [1] (22%, 53%)
8		[2, 3, 4, 8] => [6] (20%, 62%)	[1, 2, 4, 8]	[1] => [2, 4, 8] (29%, 54%)
9		[2, 3, 6, 8] => [4] (20%, 78%)		[1, 2] => [4, 8] (29%, 76%)
10		[2, 4, 6, 8] => [3] (20%, 59%)		[1, 4] => [2, 8] (29%, 71%)
11		[3, 4, 6, 8] => [2] (20%, 75%)		[1, 8] => [2, 4] (29%, 55%)
12	[2, 3, 4, 8, 10]	[2, 3] => [4, 8, 10] (21%, 51%)		[2, 4] => [1, 8] (29%, 52%)
13		[3, 10] => [2, 4, 8] (21%, 57%)		[1, 2, 4] => [8] (29%, 98%)
14		[2, 3, 4] => [8, 10] (21%, 65%)		[1, 2, 8] => [4] (29%, 77%)
15		[2, 3, 8] => [4, 10] (21%, 51%)		[1, 4, 8] => [2] (29%, 72%)
16		[2, 3, 10] => [4, 8] (21%, 78%)		[2, 4, 8] => [1] (29%, 53%)
17		[2, 4, 10] => [3, 8] (21%, 57%)	[1, 2, 6, 8]	[1, 2] => [6, 8] (23%, 61%)
18		[3, 4, 10] => [2, 8] (21%, 73%)		[1, 6] => [2, 8] (23%, 70%)
19		[3, 8, 10] => [2, 4] (21%, 58%)		[2, 6] => [1, 8] (23%, 52%)
20		[2, 3, 4, 8] => [10] (21%, 65%)		[1, 2, 6] => [8] (23%, 98%)
21		[2, 3, 4, 10] => [8] (21%, 98%)		[1, 2, 8] => [6] (23%, 62%)
22		[2, 3, 8, 10] => [4] (21%, 79%)		[1, 6, 8] => [2] (23%, 71%)
23		[2, 4, 8, 10] => [3] (21%, 58%)		[2, 6, 8] => [1] (23%, 53%)
24		[3, 4, 8, 10] => [2] (21%, 74%)	[1, 2, 8, 9]	[1, 2] => [8, 9] (22%, 60%)
25	[2, 4, 6, 8, 9]	[6, 9] => [2, 4, 8] (20%, 54%)		[1, 9] => [2, 8] (22%, 69%)
26		[2, 4, 6] => [8, 9] (20%, 59%)		[2, 9] => [1, 8] (22%, 51%)
27		[2, 4, 9] => [6, 8] (20%, 60%)		[1, 2, 8] => [9] (22%, 61%)
28		[2, 6, 9] => [4, 8] (20%, 75%)		[1, 2, 9] => [8] (22%, 98%)
29		[4, 6, 9] => [2, 8] (20%, 70%)		[1, 8, 9] => [2] (22%, 70%)
30		[6, 8, 9] => [2, 4] (20%, 54%)		[2, 8, 9] => [1] (22%, 52%)
31		[2, 4, 6, 8] => [9] (20%, 60%)	[1, 2, 8, 10]	[1, 2] => [8, 10] (25%, 65%)
32		[2, 4, 6, 9] => [8] (20%, 98%)		[1, 10] => [2, 8] (25%, 70%)
33		[2, 4, 8, 9] => [6] (20%, 61%)		[2, 10] => [1, 8] (25%, 52%)
34		[2, 6, 8, 9] => [4] (20%, 76%)		[1, 2, 8] => [10] (25%, 66%)
35		[4, 6, 8, 9] => [2] (20%, 71%)		[1, 2, 10] => [8] (25%, 98%)
36	[2, 4, 6, 8, 10]	[2, 6] => [4, 8, 10] (22%, 50%)		[1, 8, 10] => [2] (25%, 71%)
37		[6, 10] => [2, 4, 8] (22%, 54%)		[2, 8, 10] => [1] (25%, 53%)
38		[2, 4, 6] => [8, 10] (22%, 65%)	[1, 3, 4, 8]	[1, 3] => [4, 8] (23%, 76%)
39		[2, 4, 10] => [6, 8] (22%, 61%)		[1, 4] => [3, 8] (23%, 58%)
40		[2, 6, 8] => [4, 10] (22%, 51%)		[3, 4] => [1, 8] (23%, 53%)
41		[2, 6, 10] => [4, 8] (22%, 76%)		[1, 3, 4] => [8] (23%, 98%)
42		[4, 6, 10] => [2, 8] (22%, 71%)		[1, 3, 8] => [4] (23%, 77%)
43		[6, 8, 10] => [2, 4] (22%, 55%)		[1, 4, 8] => [3] (23%, 58%)
44		[2, 4, 6, 8] => [10] (22%, 66%)		[3, 4, 8] => [1] (23%, 54%)

B.7 $L^{GSAR}_{CENTRAL}$ *at site* $S_{CENTRAL}$

Column **L** represents globally frequent k-itemset, i.e., itemsets which are locally strong at all the distributed sites and column **AR(support,confidence)** shows the list of globally strong association rules, i.e., $L^{GSAR}_{CENTRAL}$ for such itemsets. Each globally strong rule has its associated support and confidence factor. The minimum threshold support is taken as 20% and minimum threshold confidence as 50%. **Site** represents the IP address of the site where the rule is locally strong. IP address 192.168.46.212 is used for site S_1, 192.168.46.189 for site S_2 and address 192.168.46.213 for site S_3.

Stong Association Rules for 2-Itemsets

S.n.	L	AR (support,confidence)	Site
1	[1, 3]	[3] => [1] (39%, 98%)	192.168.46.212
2		[3] => [1] (35%, 77%)	192.168.46.189
3		[1] => [3] (30%, 57%)	192.168.46.213
4		[3] => [1] (30%, 54%)	192.168.46.213
5	[1, 4]	[1] => [4] (62%, 63%)	192.168.46.212
6		[4] => [1] (62%, 98%)	192.168.46.212
7		[1] => [4] (40%, 52%)	192.168.46.189
8		[4] => [1] (40%, 76%)	192.168.46.189
9		[1] => [4] (40%, 76%)	192.168.46.213
10		[4] => [1] (40%, 53%)	192.168.46.213
11	[1, 7]	[1] => [7] (69%, 70%)	192.168.46.212
12		[7] => [1] (69%, 98%)	192.168.46.212
13		[1] => [7] (51%, 66%)	192.168.46.189
14		[7] => [1] (51%, 76%)	192.168.46.189
15		[7] => [1] (26%, 53%)	192.168.46.213
16	[1, 8]	[8] => [1] (33%, 98%)	192.168.46.212
17		[1] => [8] (75%, 96%)	192.168.46.189
18		[8] => [1] (75%, 77%)	192.168.46.189
19		[1] => [8] (52%, 98%)	192.168.46.213
20		[8] => [1] (52%, 53%)	192.168.46.213
21	[1, 9]	[1] => [9] (90%, 91%)	192.168.46.212
22		[9] => [1] (90%, 98%)	192.168.46.212
23		[1] => [9] (47%, 61%)	192.168.46.189
24		[9] => [1] (47%, 76%)	192.168.46.189
25		[1] => [9] (32%, 61%)	192.168.46.213
26		[9] => [1] (32%, 53%)	192.168.46.213
27	[3, 4]	[3] => [4] (25%, 64%)	192.168.46.212
28		[3] => [4] (24%, 53%)	192.168.46.189
29		[3] => [4] (44%, 77%)	192.168.46.213
30		[4] => [3] (44%, 57%)	192.168.46.213
31	[3, 7]	[3] => [7] (28%, 70%)	192.168.46.212
32		[3] => [7] (30%, 66%)	192.168.46.189
33		[7] => [3] (27%, 56%)	192.168.46.213
34	[3, 9]	[3] => [9] (36%, 91%)	192.168.46.212
35		[3] => [9] (28%, 61%)	192.168.46.189
36		[3] => [9] (34%, 61%)	192.168.46.213
37		[9] => [3] (34%, 57%)	192.168.46.213
38	[4, 7]	[4] => [7] (45%, 71%)	192.168.46.212
39		[7] => [4] (45%, 63%)	192.168.46.212
40		[4] => [7] (36%, 67%)	192.168.46.189
41		[7] => [4] (36%, 53%)	192.168.46.189
42		[7] => [4] (37%, 76%)	192.168.46.213
43	[4, 8]	[8] => [4] (22%, 64%)	192.168.46.212
44		[4] => [8] (52%, 97%)	192.168.46.189

Stong Association Rules for 3-Itemsets

S.n.	L	AR (support,confidence)	Site
1	[1, 4, 7]	[4] => [1, 7] (44%, 70%)	192.168.46.212
2		[7] => [1, 4] (44%, 62%)	192.168.46.212
3		[1, 4] => [7] (44%, 71%)	192.168.46.212
4		[1, 7] => [4] (44%, 63%)	192.168.46.212
5		[4, 7] => [1] (44%, 98%)	192.168.46.212
6		[4] => [1, 7] (27%, 51%)	192.168.46.189
7		[1, 4] => [7] (27%, 67%)	192.168.46.189
8		[1, 7] => [4] (27%, 53%)	192.168.46.189
9		[4, 7] => [1] (27%, 76%)	192.168.46.189
10		[1, 7] => [4] (20%, 76%)	192.168.46.213
11		[4, 7] => [1] (20%, 53%)	192.168.46.213
12	[1, 4, 8]	[8] => [1, 4] (21%, 63%)	192.168.46.212
13		[1, 8] => [4] (21%, 64%)	192.168.46.212
14		[4, 8] => [1] (21%, 98%)	192.168.46.212
15		[1] => [4, 8] (39%, 51%)	192.168.46.189
16		[4] => [1, 8] (39%, 74%)	192.168.46.189
17		[1, 4] => [8] (39%, 97%)	192.168.46.189
18		[1, 8] => [4] (39%, 53%)	192.168.46.189
19		[4, 8] => [1] (39%, 76%)	192.168.46.189
20		[1] => [4, 8] (40%, 75%)	192.168.46.213
21		[4] => [1, 8] (40%, 52%)	192.168.46.213
22		[1, 4] => [8] (40%, 98%)	192.168.46.213
23		[1, 8] => [4] (40%, 76%)	192.168.46.213
24		[4, 8] => [1] (40%, 53%)	192.168.46.213
25	[1, 4, 9]	[1] => [4, 9] (57%, 58%)	192.168.46.212
26		[4] => [1, 9] (57%, 90%)	192.168.46.212
27		[9] => [1, 4] (57%, 62%)	192.168.46.212
28		[1, 4] => [9] (57%, 91%)	192.168.46.212
29		[1, 9] => [4] (57%, 63%)	192.168.46.212
30		[4, 9] => [1] (57%, 98%)	192.168.46.212
31		[1, 4] => [9] (25%, 62%)	192.168.46.189
32		[1, 9] => [4] (25%, 53%)	192.168.46.189
33		[4, 9] => [1] (25%, 75%)	192.168.46.189
34		[1, 4] => [9] (25%, 61%)	192.168.46.213
35		[1, 9] => [4] (25%, 76%)	192.168.46.213
36		[4, 9] => [1] (25%, 53%)	192.168.46.213
37	[1, 7, 8]	[8] => [1, 7] (24%, 70%)	192.168.46.212
38		[1, 8] => [7] (24%, 71%)	192.168.46.212
39		[7, 8] => [1] (24%, 98%)	192.168.46.212
40		[1] => [7, 8] (49%, 64%)	192.168.46.189
41		[7] => [1, 8] (49%, 74%)	192.168.46.189
42		[8] => [1, 7] (49%, 51%)	192.168.46.189
43		[1, 7] => [8] (49%, 96%)	192.168.46.189
44		[1, 8] => [7] (49%, 66%)	192.168.46.189

Appendix C - Datasets used for AeQARM-AAPDB System

C.1 PDB_1 *at site* S_1

This is a real dataset of protein sequences, the Astral Structural Classification of Proteins (SCOP) [75, 84] version 1.75 genetic domain sequence subsets, based on PDB SEQRES records with less than 40% identity to each other. A comprehensive ordering of all proteins of known structure according to their evolutionary and structural relationships is done in SCOP database. Protein domains here are grouped into species and classified hierarchically into families, superfamilies, folds and classes. In this dataset each protein record starts with greater than ('>') character followed by protein description headers according to SCOP. An additional hash ('#') character is inserted as a separator character between protein description headers and protein sequence of amino acids. A total of 3523 protein records are stored in this protein data bank (PDB_1) at site S_1.

```
>d1dlwa_ a.1.1.1 (A:) Protozoan/bacterial hemoglobin (Ciliate (Paramecium caudatum) [TaxId: 5885])#slfeqlg
>d1s69a_ a.1.1.1 (A:) Protozoan/bacterial hemoglobin (Cyanobacteria (Synechocystis sp.), pcc 6803 [TaxId:
>d1idra_ a.1.1.1 (A:) Protozoan/bacterial hemoglobin (Mycobacterium tuberculosis, HbN [TaxId: 1773])#gllsr
>d1ngka_ a.1.1.1 (A:) Protozoan/bacterial hemoglobin (Mycobacterium tuberculosis, HbO [TaxId: 1773])#ksfyd
>d1ux8a_ a.1.1.1 (A:) Protozoan/bacterial hemoglobin (Bacillus subtilis [TaxId: 1423])#napyeaigeellsqlvdtf
>d1kr7a_ a.1.1.4 (A:) Nerve tissue mini-hemoglobin (neural globin) (Milky ribbon-worm (Cerebratulus lacteu
>d3sdha_ a.1.1.2 (A:) Hemoglobin I (Ark clam (Scapharca inaequivalvis) [TaxId: 6561])#svydaaaqltadvkkdlrds
>d1b0ba_ a.1.1.2 (A:) Hemoglobin I (Clam (Lucina pectinata) [TaxId: 29163])#slsaaqkdnvksswakasaawgtagpeffm
>d1h97a_ a.1.1.2 (A:) Trematode hemoglobin/myoglobin (Paramphistomum epiclitum [TaxId: 54403])#tltkheqdill
>d1jl7a_ a.1.1.2 (A:) Glycera globin (Marine bloodworm (Glycera dibranchiata) [TaxId: 6350])#glsaaqrqvvast
>d1a6ma_ a.1.1.2 (A:) Myoglobin (Sperm whale (Physeter catodon) [TaxId: 9755])#vlsegewqlvlhvwakveadvaghgqd
>d1mbaa_ a.1.1.2 (A:) Myoglobin (Slug sea hare (Aplysia limacina) [TaxId: 6502])#slsaaeadlagkswapvfanknang
>d1ecda_ a.1.1.2 (A:) Erythrocruorin (Midge (Chironomus thummi thummi), fraction III [TaxId: 7154])#lsadqi
>d2gdma_ a.1.1.2 (A:) Leghemoglobin (Yellow lupin (Lupinus luteus) [TaxId: 3873])#galtesqaalvkssweefnanipkl
>d1cg5a_ a.1.1.2 (A:) Hemoglobin, alpha-chain (Cartilaginous fish akaei (Dasyatis akajei) [TaxId: 31902])#
>d3d1ka1 a.1.1.2 (A:1-142) Hemoglobin, alpha-chain (Antarctic fish (Trematomus newnesi) [TaxId: 35730])#sl
>d1cg5b_ a.1.1.2 (B:) Hemoglobin, beta-chain (Cartilaginous fish akaei (Dasyatis akajei) [TaxId: 31902])#v
>d3d1kb1 a.1.1.2 (B:1-146) Hemoglobin, beta-chain (Antarctic fish (Trematomus newnesi) [TaxId: 35730])#vew
>d1gcvb_ a.1.1.2 (B:) Hemoglobin, beta-chain (Houndshark (Mustelus griseus) [TaxId: 89020])#vhwtqeerdeiskt
>d1it2a_ a.1.1.2 (A:) Hagfish hemoglobin (Inshore hagfish (Eptatretus burgeri) [TaxId: 7764])#piidqgplptlt
>d1asha_ a.1.1.2 (A:) Ascaris hemoglobin, domain 1 (Pig roundworm (Ascaris suum) [TaxId: 6253])#anktrelcmk
>d1itha_ a.1.1.2 (A:) Hemoglobin (Innkeeper worm (Urechis caupo) [TaxId: 6431])#gltaaqikaiqdhwflnikgclqaaa
>d1hlba_ a.1.1.2 (A:) Hemoglobin, different isoforms (Caudina arenicola, also known as Molpadia arenicola
>d1q1fa_ a.1.1.2 (A:) Neuroglobin (Mouse (Mus musculus) [TaxId: 10090])#rpeselirqswrvvsrsplehgtvlfarlfalep
>d1cqxa1 a.1.1.2 (A:1-150) Flavohemoglobin, N-terminal domain (Alcaligenes eutrophus [TaxId: 106590])#mltq
>d2qfka1 a.1.1.2 (A:1-137) Dehaloperoxidase (Amphitrite ornata [TaxId: 129555])#gfkqdiatirgdlrtyaqdiflafln
>d1or4a_ a.1.1.2 (A:) Heme-based aerotactic transducer HemAT, sensor domain (Bacillus subtilis [TaxId: 142
>d1urva_ a.1.1.2 (A:) Cytoglobin (Human (Homo sapiens) [TaxId: 9606])#elseaerkavqamwarlyansedvgvailvrffvnf
```

C.2 PDB_2 *at site* S_2

A total of 3523 protein records are stored in this protein data bank (PDB_2) at site S_2.

```
>d1kqfa1 b.52.2.2 (A:851-1015) Formate dehydrogenase N, alpha subunit (Escherichia coli [TaxId: 562])#epie
>d1g8ka1 b.52.2.2 (A:683-825) Arsenite oxidase large subunit (Alcaligenes faecalis [TaxId: 511])#lpatvqqqk
>d2jioa1 b.52.2.2 (A:601-723) Periplasmic nitrate reductase alpha chain, NapA (Desulfovibrio desulfuricans
>d1ogya1 b.52.2.2 (A:682-801) Periplasmic nitrate reductase alpha chain, NapA (Rhodobacter sphaeroides [Ta
>d1y5ia1 b.52.2.2 (A:1075-1244) Respiratory nitrate reductase 1 alpha chain (Escherichia coli [TaxId: 562]
>d1vlfm1 b.52.2.2 (M:729-875) Transhydroxylase alpha subunit, AthL (Pelobacter acidigallici [TaxId: 35816]
>d2fug31 b.52.2.2 (3:686-767) NADH-quinone oxidoreductase chain 3, Nqo3, C-terminal domain (Thermus thermo
>d1qcsa1 b.52.2.3 (A:0-85) N-terminal domain of NSF-N, NSF-Nn (Hamster (Cricetulus griseus) [TaxId: 10029]
>d1cr5a1 b.52.2.3 (A:26-107) N-terminal domain of NSF-N, NSF-Nn (Baker's yeast (Saccharomyces cerevisiae),
>d1cz5a1 b.52.2.3 (A:1-91) N-terminal domain of VAT-N, VAT-Nn (Archaeon Thermoplasma acidophilum [TaxId: 2
>d1e32a1 b.52.2.3 (A:21-106) Membrane fusion ATPase p97 N-terminal domain , P97-Nn (Mouse (Mus musculus) [
>d1w1fa2 b.52.2.3 (A:13-99) Peroxisome biogenesis factor 1 (PEX-1), N-terminal domain (Mouse (Mus musculus
>d1dfup_ b.53.1.1 (P:) Ribosomal protein L25 (Escherichia coli [TaxId: 562])#mftinaevrkeqgkgasrrlraankfpai
>d1feua_ b.53.1.1 (A:) Ribosomal protein TL5 (general stress protein CTC) (Thermus thermophilus [TaxId: 27
>d2zjrs1 b.53.1.1 (S:1-175) Ribosomal protein TL5 (general stress protein CTC) (Deinococcus radiodurans [T
>d1gtra1 b.53.1.2 (A:339-547) Gln-tRNA synthetase (GlnRS), C-terminal (anticodon-binding) domain (Escheric
>d1e50b_ b.54.1.1 (B:) Core binding factor beta, CBF (Human (Homo sapiens) [TaxId: 9606])#prvvpdqrskfeneef
>d2jn4a1 b.173.1.1 (A:1-66) Uncharacterized protein FixU (NifT) (Rhodopseudomonas palustris [TaxId: 1076])
>d1maia_ b.55.1.1 (A:) Phospholipase C delta-1 (Rat (Rattus norvegicus) [TaxId: 10116])#glqddpdlqallkgsqlll
>d1btna_ b.55.1.1 (A:) beta-spectrin (Mouse (Mus musculus), brain [TaxId: 10090])#megflnrkheweahnkkassrswh
>d1droa_ b.55.1.1 (A:) beta-spectrin (Fruit fly (Drosophila melanogaster) [TaxId: 7227])#gsgtgageghegyvtrkl
>d2dyna_ b.55.1.1 (A:) Dynamin (Human (Homo sapiens) [TaxId: 9606])#ilvirkgwltinnigimkggskeywfvltaenlswykd
>d1btka_ b.55.1.1 (A:) Bruton's tyrosine kinase (Human (Homo sapiens) [TaxId: 9606])#aavilesiflkrsqqkkktsp
>d1plsa_ b.55.1.1 (A:) Pleckstrin (Human (Homo sapiens) [TaxId: 9606])#mepkriregylvkkgsvfntwkpmwvvlledgief
>d2i5fa1 b.55.1.1 (A:244-347) Pleckstrin (Human (Homo sapiens) [TaxId: 9606])#viikqgcllkqghrrknwkvrkfilred
>d1dbha2 b.55.1.1 (A:418-550) Son of sevenless-1 (sos-1) (Human (Homo sapiens) [TaxId: 9606])#aikkmneiqkni
>d1foea2 b.55.1.1 (A:1240-1401) GEF of TIAM1 (T-Lymphoma invasion and metastasis inducing protein 1) (Mous
>d1kz7a2 b.55.1.1 (A:819-965) Dbl's big sister, Dbs (Mouse (Mus musculus) [TaxId: 10090])#tgydgnlgdlgkllmq
```

C.3 PDB_3 *at site* S_3

A total of 3523 protein records are stored in this protein data bank (PDB_3) at site S_3.

```
>d2fkia1 d.198.3.1 (A:1-118) Hypothetical protein YjbR (Escherichia coli [TaxId: 562])#mtisellqycmakpgaeqs
>d2i8da1 d.198.4.1 (A:2-122) Uncharacterized protein LSEI2283 (Lactobacillus casei [TaxId: 1582])#gslaewyq
>d2g7ja1 d.198.5.1 (A:1-112) Putative cytoplasmic protein YgaC (Salmonella typhimurium [TaxId: 90371])#myl
>d2od0a1 d.198.5.2 (A:3-105) Hypothetical protein VP1028 (Vibrio parahaemolyticus [TaxId: 670])#kpilkdsmkl
>d1i3ja_ d.285.1.1 (A:) DNA-binding domain of intron endonuclease I-TevI (Bacteriophage T4 [TaxId: 10665])
>d1u3em2 d.285.1.1 (M:106-174) Intron-encoded homing endonuclease I-HmuI (Bacteriophage SP01 [TaxId: 10685
>d1kafa_ d.199.1.1 (A:) DNA-binding C-terminal domain of the transcription factor MotA (Bacteriophage T4 [
>d2a8ea1 d.296.1.1 (A:2-211) Hypothetical protein YktB (Bacillus subtilis [TaxId: 1423])#tqmrfteedfntftieg
>d1yb3a1 d.296.1.2 (A:2-167) Hypothetical protein PF0168 (Pyrococcus furiosus [TaxId: 2261])#mlkevhellnriw
>d1rk8b_ d.232.1.1 (B:) Mago nashi protein (Fruit fly (Drosophila melanogaster) [TaxId: 7227])#edfylryyvgh
>d2cr9a1 d.297.1.1 (A:8-133) Poly [ADP-ribose] polymerase-1, PARP-1 (Human (Homo sapiens) [TaxId: 9606])#k
>d2hq4a1 d.342.1.1 (A:1-152) Hypothetical protein PH1570 (Pyrococcus horikoshii [TaxId: 53953])#mqceeklevf
>d2dy1a3 d.14.1.1 (A:455-569) Elongation factor G (EF-G), domain IV (Thermus thermophilus, EF-G-2 [TaxId:
>d1n0ua3 d.14.1.1 (A:561-725) Elongation factor 2 (eEF-2), domain IV (Baker's yeast (Saccharomyces cerevis
>d1pkpa1 d.14.1.1 (A:78-148) Ribosomal protein S5, C-terminal domain (Bacillus stearothermophilus [TaxId:
>d2gy9i1 d.14.1.1 (I:4-129) Ribosomal protein S9 (Escherichia coli [TaxId: 562])#qyygtgrrkssaarvfikpgngkiv
>d1a6fa_ d.14.1.2 (A:) RNase P protein (Bacillus subtilis [TaxId: 1423])#ahlkkrnrlkknedfqkvfkhgtsvanrqfvly
>d1nz0a_ d.14.1.2 (A:) RNase P protein (Thermotoga maritima [TaxId: 2336])#erlrlrrdfllifkegkslqneyfvvlfrkn
>d1b63a1 d.14.1.3 (A:217-331) DNA mismatch repair protein MutL (Escherichia coli [TaxId: 562])#gtafleqalai
>d1h7sa1 d.14.1.3 (A:232-365) DNA mismatch repair protein PMS2 (Human (Homo sapiens) [TaxId: 9606])#gqkqlq
>d1ei1a1 d.14.1.3 (A:221-392) DNA gyrase B (Escherichia coli [TaxId: 562])#gikafveylnknktpihpnifyfstekdgig
>d1pvga1 d.14.1.3 (A:246-406) DNA topoisomerase II (Baker's yeast (Saccharomyces cerevisiae) [TaxId: 4932]
>d1s16a1 d.14.1.3 (A:1217-1383) Topoisomerase IV subunit B (Escherichia coli [TaxId: 562])#dglndylaeavnglp
>d2hkja2 d.14.1.3 (A:307-470) Topoisomerase VI-B subunit (Archaeon Sulfolobus shibatae [TaxId: 2286])#rsps
>d1usua_ d.14.1.8 (A:) Heat shock protein hsp82 (Baker's yeast (Saccharomyces cerevisiae) [TaxId: 4932])#p
>d1r6la1 d.14.1.4 (A:1-151) Ribonuclease PH, domain 1 (Pseudomonas aeruginosa [TaxId: 287])#mnrpsgraadqlrp
>d1e3ha2 d.14.1.4 (A:3-151) Polynucleotide phosphorylase/guanosine pentaphosphate synthase (PNPase/GPSI),
>d1e3ha3 d.14.1.4 (A:346-482) Polynucleotide phosphorylase/guanosine pentaphosphate synthase (PNPase/GPSI),
```

C.4 $FPDB_1$ *at site* S_1

PDB_1 shown in Appendix C.1 is further filtered to generate $FPDB_1$ for the protein sequence length range ≥ 50 and < 400. A total of 3341 such filtered protein records are obtained.

```
>d1dlwa_ a.1.1.1 (A:) Protozoan/bacterial hemoglobin (Ciliate (Paramecium caudatum) [TaxId: 5885])#slfeqlg
>d1s69a_ a.1.1.1 (A:) Protozoan/bacterial hemoglobin (Cyanobacteria (Synechocystis sp.), pcc 6803 [TaxId:
>d1idra_ a.1.1.1 (A:) Protozoan/bacterial hemoglobin (Mycobacterium tuberculosis, HbN [TaxId: 1773])#gllsr
>d1ngka_ a.1.1.1 (A:) Protozoan/bacterial hemoglobin (Mycobacterium tuberculosis, HbO [TaxId: 1773])#ksfyd
>d1ux8a_ a.1.1.1 (A:) Protozoan/bacterial hemoglobin (Bacillus subtilis [TaxId: 1423])#napyeaigeellsqlvdtf
>d1kr7a_ a.1.1.4 (A:) Nerve tissue mini-hemoglobin (neural globin) (Milky ribbon-worm (Cerebratulus lacteu
>d3sdha_ a.1.1.2 (A:) Hemoglobin I (Ark clam (Scapharca inaequivalvis) [TaxId: 6561])#svydaaaqltadvkkdlrds
>d1b0ba_ a.1.1.2 (A:) Hemoglobin I (Clam (Lucina pectinata) [TaxId: 29163])#slsaaqkdnvksswakasaawgtagpeffm
>d1h97a_ a.1.1.2 (A:) Trematode hemoglobin/myoglobin (Paramphistomum epiclitum [TaxId: 54403])#tltkheqdill
>d1jl7a_ a.1.1.2 (A:) Glycera globin (Marine bloodworm (Glycera dibranchiata) [TaxId: 6350])#glsaaqrqvvast
>d1a6ma_ a.1.1.2 (A:) Myoglobin (Sperm whale (Physeter catodon) [TaxId: 9755])#vlsegewqlvlhvwakveadvaghgqd
>d1mbaa_ a.1.1.2 (A:) Myoglobin (Slug sea hare (Aplysia limacina) [TaxId: 6502])#slsaaeadlagkswapvfanknang
>d1ecda_ a.1.1.2 (A:) Erythrocruorin (Midge (Chironomus thummi thummi), fraction III [TaxId: 7154])#lsadqi
>d2gdma_ a.1.1.2 (A:) Leghemoglobin (Yellow lupin (Lupinus luteus) [TaxId: 3873])#galtesqaalvkssweefnanipk
>d1cg5a_ a.1.1.2 (A:) Hemoglobin, alpha-chain (Cartilaginous fish akaei (Dasyatis akajei) [TaxId: 31902])#
>d3d1ka1 a.1.1.2 (A:1-142) Hemoglobin, alpha-chain (Antarctic fish (Trematomus newnesi) [TaxId: 35730])#sl
>d1cg5b_ a.1.1.2 (B:) Hemoglobin, beta-chain (Cartilaginous fish akaei (Dasyatis akajei) [TaxId: 31902])#v
>d3d1kb1 a.1.1.2 (B:1-146) Hemoglobin, beta-chain (Antarctic fish (Trematomus newnesi) [TaxId: 35730])#vew
>d1gcvb_ a.1.1.2 (B:) Hemoglobin, beta-chain (Houndshark (Mustelus griseus) [TaxId: 89020])#vhwtqeerdeiskt
>d1it2a_ a.1.1.2 (A:) Hagfish hemoglobin (Inshore hagfish (Eptatretus burgeri) [TaxId: 7764])#piidqgplptlt
>d1asha_ a.1.1.2 (A:) Ascaris hemoglobin, domain 1 (Pig roundworm (Ascaris suum) [TaxId: 6253])#anktrelcmk
>d1itha_ a.1.1.2 (A:) Hemoglobin (Innkeeper worm (Urechis caupo) [TaxId: 6431])#gltaaqikaiqdhwflnikgclqaaa
>d1hlba_ a.1.1.2 (A:) Hemoglobin, different isoforms (Caudina arenicola, also known as Molpadia arenicola
>d1q1fa_ a.1.1.2 (A:) Neuroglobin (Mouse (Mus musculus) [TaxId: 10090])#rpeselirqswrvvsrsplehgtvlfarlfalep
>d1cqxa1 a.1.1.2 (A:1-150) Flavohemoglobin, N-terminal domain (Alcaligenes eutrophus [TaxId: 106590])#mltq
>d2qfka1 a.1.1.2 (A:1-137) Dehaloperoxidase (Amphitrite ornata [TaxId: 129555])#gfkqdiatirgdlrtyaqdiflafln
>d1or4a_ a.1.1.2 (A:) Heme-based aerotactic transducer HemAT, sensor domain (Bacillus subtilis [TaxId: 142
>d1urva_ a.1.1.2 (A:) Cytoglobin (Human (Homo sapiens) [TaxId: 9606])#elseaerkavqamwarlyansedvgvailvrffvnf
```

C.5 $FPDB_2$ *at site* S_2

PDB_2 shown in Appendix C.2 is further filtered to generate $FPDB_2$ for the protein sequence length range ≥ 50 and < 400. A total of 3253 such filtered protein records are obtained.

```
>d1kqfa1 b.52.2.2 (A:851-1015) Formate dehydrogenase N, alpha subunit (Escherichia coli [TaxId: 562])#epie
>d1g8ka1 b.52.2.2 (A:683-825) Arsenite oxidase large subunit (Alcaligenes faecalis [TaxId: 511])#lpatvqqqk
>d2jioa1 b.52.2.2 (A:601-723) Periplasmic nitrate reductase alpha chain, NapA (Desulfovibrio desulfuricans
>d1ogya1 b.52.2.2 (A:682-801) Periplasmic nitrate reductase alpha chain, NapA (Rhodobacter sphaeroides [Ta
>d1y5ia1 b.52.2.2 (A:1075-1244) Respiratory nitrate reductase 1 alpha chain (Escherichia coli [TaxId: 562]
>d1vlfm1 b.52.2.2 (M:729-875) Transhydroxylase alpha subunit, AthL (Pelobacter acidigallici [TaxId: 35816]
>d2fug31 b.52.2.2 (3:686-767) NADH-quinone oxidoreductase chain 3, Nqo3, C-terminal domain (Thermus thermo
>d1qcsa1 b.52.2.3 (A:0-85) N-terminal domain of NSF-N, NSF-Nn (Hamster (Cricetulus griseus) [TaxId: 10029]
>d1cr5a1 b.52.2.3 (A:26-107) N-terminal domain of NSF-N, NSF-Nn (Baker's yeast (Saccharomyces cerevisiae),
>d1cz5a1 b.52.2.3 (A:1-91) N-terminal domain of VAT-N, VAT-Nn (Archaeon Thermoplasma acidophilum [TaxId: 2
>d1e32a1 b.52.2.3 (A:21-106) Membrane fusion ATPase p97 N-terminal domain , P97-Nn (Mouse (Mus musculus) [
>d1w1fa2 b.52.2.3 (A:13-99) Peroxisome biogenesis factor 1 (PEX-1), N-terminal domain (Mouse (Mus musculus
>d1dfup_ b.53.1.1 (P:) Ribosomal protein L25 (Escherichia coli [TaxId: 562])#mftinaevrkeqgkgasrrlraankfpai
>d1feua_ b.53.1.1 (A:) Ribosomal protein TL5 (general stress protein CTC) (Thermus thermophilus [TaxId: 27
>d2zjrs1 b.53.1.1 (S:1-175) Ribosomal protein TL5 (general stress protein CTC) (Deinococcus radiodurans [T
>d1gtra1 b.53.1.2 (A:339-547) Gln-tRNA synthetase (GlnRS), C-terminal (anticodon-binding) domain (Escheric
>d1e50b_ b.54.1.1 (B:) Core binding factor beta, CBF (Human (Homo sapiens) [TaxId: 9606])#prvvpdqrskfeneef
>d2jn4a1 b.173.1.1 (A:1-66) Uncharacterized protein FixU (NifT) (Rhodopseudomonas palustris [TaxId: 1076])
>d1maia_ b.55.1.1 (A:) Phospholipase C delta-1 (Rat (Rattus norvegicus) [TaxId: 10116])#glqddpdlqallkgsqll
>d1btna_ b.55.1.1 (A:) beta-spectrin (Mouse (Mus musculus), brain [TaxId: 10090])#megflnrkheweahnkkassrswh
>d1droa_ b.55.1.1 (A:) beta-spectrin (Fruit fly (Drosophila melanogaster) [TaxId: 7227])#gsgtgageghegyvtrk
>d2dyna_ b.55.1.1 (A:) Dynamin (Human (Homo sapiens) [TaxId: 9606])#ilvirkgwltinnigimkggskeywfvltaenlswykd
>d1btka_ b.55.1.1 (A:) Bruton's tyrosine kinase (Human (Homo sapiens) [TaxId: 9606])#aavilesiflkrsqqkkktsp
>d1plsa_ b.55.1.1 (A:) Pleckstrin (Human (Homo sapiens) [TaxId: 9606])#mepkriregylvkkgsvfntwkpmwvvlledgief
>d2i5fa1 b.55.1.1 (A:244-347) Pleckstrin (Human (Homo sapiens) [TaxId: 9606])#viikqgcllkqghrrknwkvrkfilred
>d1dbha2 b.55.1.1 (A:418-550) Son of sevenless-1 (sos-1) (Human (Homo sapiens) [TaxId: 9606])#aikkmneiqkni
>d1foea2 b.55.1.1 (A:1240-1401) GEF of TIAM1 (T-Lymphoma invasion and metastasis inducing protein 1) (Mous
>d1kz7a2 b.55.1.1 (A:819-965) Dbl's big sister, Dbs (Mouse (Mus musculus) [TaxId: 10090])#tgydgnlgdlgkllmq
```

C.6 $FPDB_3$ *at site* S_3

PDB_3 shown in Appendix C.3 is further filtered to generate $FPDB_3$ for the protein sequence length range ≥ 50 and < 400. A total of 3039 such filtered protein records are obtained.

```
>d2fkia1 d.198.3.1 (A:1-118) Hypothetical protein YjbR (Escherichia coli [TaxId: 562])#mtisellqycmakpgaeqs
>d2i8da1 d.198.4.1 (A:2-122) Uncharacterized protein LSEI2283 (Lactobacillus casei [TaxId: 1582])#gslaewyq
>d2g7ja1 d.198.5.1 (A:1-112) Putative cytoplasmic protein YgaC (Salmonella typhimurium [TaxId: 90371])#myl
>d2od0a1 d.198.5.2 (A:3-105) Hypothetical protein VP1028 (Vibrio parahaemolyticus [TaxId: 670])#kpilkdsmkl
>d1i3ja_ d.285.1.1 (A:) DNA-binding domain of intron endonuclease I-TevI (Bacteriophage T4 [TaxId: 10665])
>d1u3em2 d.285.1.1 (M:106-174) Intron-encoded homing endonuclease I-HmuI (Bacteriophage SP01 [TaxId: 10685
>d1kafa_ d.199.1.1 (A:) DNA-binding C-terminal domain of the transcription factor MotA (Bacteriophage T4 [
>d2a8ea1 d.296.1.1 (A:2-211) Hypothetical protein YktB (Bacillus subtilis [TaxId: 1423])#tqmrfteedfntftieg
>d1yb3a1 d.296.1.2 (A:2-167) Hypothetical protein PF0168 (Pyrococcus furiosus [TaxId: 2261])#mlkevhellnriw
>d1rk8b_ d.232.1.1 (B:) Mago nashi protein (Fruit fly (Drosophila melanogaster) [TaxId: 7227])#edfylryyvghl
>d2cr9a1 d.297.1.1 (A:8-133) Poly [ADP-ribose] polymerase-1, PARP-1 (Human (Homo sapiens) [TaxId: 9606])#k
>d2hq4a1 d.342.1.1 (A:1-152) Hypothetical protein PH1570 (Pyrococcus horikoshii [TaxId: 53953])#mqceeklevf
>d2dy1a3 d.14.1.1 (A:455-569) Elongation factor G (EF-G), domain IV (Thermus thermophilus, EF-G-2 [TaxId:
>d1n0ua3 d.14.1.1 (A:561-725) Elongation factor 2 (eEF-2), domain IV (Baker's yeast (Saccharomyces cerevis
>d1pkpa1 d.14.1.1 (A:78-148) Ribosomal protein S5, C-terminal domain (Bacillus stearothermophilus [TaxId:
>d2gy9i1 d.14.1.1 (I:4-129) Ribosomal protein S9 (Escherichia coli [TaxId: 562])#qyygtgrrkssaarvfikpgngkiv
>d1a6fa_ d.14.1.2 (A:) RNase P protein (Bacillus subtilis [TaxId: 1423])#ahlkkrnrlkknedfqkvfkhgtsvanrqfvly
>d1nz0a_ d.14.1.2 (A:) RNase P protein (Thermotoga maritima [TaxId: 2336])#erlrlrrdfllifkegkslqneyfvvlfrkn
>d1b63a1 d.14.1.3 (A:217-331) DNA mismatch repair protein MutL (Escherichia coli [TaxId: 562])#gtafleqalai
>d1h7sa1 d.14.1.3 (A:232-365) DNA mismatch repair protein PMS2 (Human (Homo sapiens) [TaxId: 9606])#gqkqlq
>d1ei1a1 d.14.1.3 (A:221-392) DNA gyrase B (Escherichia coli [TaxId: 562])#gikafveylnknktpihpnifyfstekdgig
>d1pvga1 d.14.1.3 (A:246-406) DNA topoisomerase II (Baker's yeast (Saccharomyces cerevisiae) [TaxId: 4932]
>d1s16a1 d.14.1.3 (A:1217-1383) Topoisomerase IV subunit B (Escherichia coli [TaxId: 562])#dglndylaeavnglp
>d2hkja2 d.14.1.3 (A:307-470) Topoisomerase VI-B subunit (Archaeon Sulfolobus shibatae [TaxId: 2286])#rsps
>d1usua_ d.14.1.8 (A:) Heat shock protein hsp82 (Baker's yeast (Saccharomyces cerevisiae) [TaxId: 4932])#p
>d1r6la1 d.14.1.4 (A:1-151) Ribonuclease PH, domain 1 (Pseudomonas aeruginosa [TaxId: 287])#mnrpsgraadqlrp
>d1e3ha2 d.14.1.4 (A:3-151) Polynucleotide phosphorylase/guanosine pentaphosphate synthase (PNPase/GPSI),
>d1e3ha3 d.14.1.4 (A:346-482) Polynucleotide phosphorylase/guanosine pentaphosphate synthase (PNPase/GPSI),
```

C.7 AAF_1 *generated from* $FPDB_1$ *at site* S_1

Each column heading represents single letter code of an amino acid. Each row represents the frequencies of amino acids in a particular protein sequence record.

S. N.	a	c	d	e	f	g	h	i	k	l	m	n	p	q	r	s	t	v	w	y
1)	22	1	3	4	6	11	2	3	2	8	2	8	2	8	3	3	12	14	1	1
2)	17	0	12	10	6	9	6	2	9	12	3	3	2	4	6	1	6	10	0	5
3)	14	0	6	8	6	13	5	8	5	12	3	1	6	4	6	8	7	12	0	3
4)	13	1	10	12	6	6	5	4	2	12	4	1	5	3	15	8	4	7	2	6
5)	9	1	5	12	6	7	6	4	4	16	4	3	9	5	8	4	7	4	1	4
6)	21	2	7	3	5	12	6	4	9	8	1	6	2	3	2	6	2	6	1	4
7)	18	1	12	3	6	10	2	8	15	13	3	10	2	6	4	7	6	13	2	4
8)	28	1	8	5	10	16	2	2	11	9	6	6	3	5	2	11	5	7	4	1
9)	14	0	6	14	7	8	11	10	14	11	5	3	5	5	3	6	11	9	0	5
10)	27	1	6	8	4	21	5	7	11	10	5	3	3	4	3	12	1	11	2	3
11)	17	0	7	14	6	10	12	9	19	18	2	1	4	4	4	6	5	8	2	3
12)	29	0	8	5	15	11	1	4	11	11	3	9	6	2	4	13	2	10	2	0
13)	17	0	9	5	14	11	4	9	10	6	4	5	5	4	3	9	9	9	1	2
14)	21	0	6	14	7	7	5	9	14	14	1	6	5	4	1	9	8	17	3	2
15)	17	4	7	10	8	5	10	3	10	20	1	9	3	4	4	6	6	10	1	3
16)	13	1	9	5	5	8	5	10	14	14	4	4	7	2	5	14	5	11	2	4
17)	12	0	8	9	9	7	7	6	15	14	1	5	2	10	5	6	6	12	2	5
18)	15	2	12	5	7	10	7	9	9	13	5	7	3	4	4	10	7	9	2	6
19)	7	2	15	5	9	5	8	9	12	9	3	1	3	7	6	6	10	12	3	4
20)	6	2	8	10	10	4	2	17	18	13	1	7	7	8	2	13	5	7	2	4

C.8 AAF_2 *generated from* $FPDB_2$ *at site* S_2

Each column heading represents single letter code of an amino acid. Each row represents the frequencies of amino acids in a particular protein sequence record.

S. N.	a	c	d	e	f	g	h	i	k	l	m	n	p	q	r	s	t	v	w	y
1)	14	0	3	12	6	12	5	9	9	12	1	12	11	7	9	5	14	17	2	5
2)	5	1	12	6	8	12	1	7	8	5	6	8	5	9	9	6	10	11	4	10
3)	15	2	9	11	4	4	3	7	8	8	3	3	10	1	9	4	7	11	1	3
4)	7	2	6	9	7	9	3	5	5	8	3	4	7	2	15	7	6	10	4	1
5)	11	0	9	8	4	19	6	9	7	12	6	13	6	9	8	10	10	13	4	6
6)	11	4	11	7	1	14	4	11	9	8	6	8	6	2	8	9	6	8	2	12
7)	17	0	1	9	2	6	4	0	5	6	1	0	6	3	7	1	3	7	2	2
8)	7	2	2	4	3	5	3	4	3	6	2	3	5	5	5	11	4	8	1	3
9)	6	1	5	0	6	5	2	6	3	5	0	12	5	3	3	5	5	5	2	3
10)	5	1	7	8	0	7	0	7	6	4	3	6	2	0	13	7	2	12	0	1
11)	2	3	8	5	1	3	0	6	5	9	2	7	4	3	10	6	2	10	0	0
12)	7	2	4	4	4	6	5	2	1	10	0	5	3	8	6	6	3	9	1	1
13)	11	0	5	7	4	5	3	7	11	5	3	3	4	5	6	2	2	8	0	3
14)	11	0	11	18	6	11	6	8	9	21	3	4	15	4	16	9	3	24	0	6
15)	14	1	11	11	6	12	4	9	10	12	5	5	13	8	10	6	13	22	0	3
16)	20	2	14	17	10	13	4	11	14	13	4	9	13	5	15	10	8	19	2	6
17)	8	4	7	16	9	9	1	5	6	9	2	4	6	10	15	6	4	7	3	3
18)	9	0	2	6	0	4	1	3	6	8	4	2	4	1	2	1	5	5	2	1
19)	5	2	10	9	5	5	3	8	9	12	2	1	5	10	11	10	3	5	3	1
20)	10	2	5	10	4	4	5	5	12	6	3	7	2	3	3	10	1	6	3	5

⋮

C.9 AAF_3 *generated from* $FPDB_3$ *at site* S_3

Each column heading represents single letter code of an amino acid. Each row represents the frequencies of amino acids in a particular protein sequence record.

S. N.	a	c	d	e	f	g	h	i	k	l	m	n	p	q	r	s	t	v	w	y
1)	9	1	6	9	1	2	4	3	8	16	3	4	6	9	5	10	4	11	2	5
2)	8	0	9	5	9	4	5	10	8	9	4	3	9	11	5	4	9	2	4	3
3)	7	1	7	7	4	10	5	5	4	10	2	5	5	1	11	6	6	7	0	9
4)	7	0	5	7	7	6	1	5	11	11	3	3	3	4	4	10	5	6	1	4
5)	5	6	5	2	6	5	3	8	15	1	1	7	1	1	5	12	4	4	2	3
6)	3	2	2	4	1	6	3	7	10	5	0	3	3	3	3	3	4	4	0	3
7)	5	1	6	10	4	6	2	12	11	9	7	11	0	1	5	5	4	5	0	4
8)	14	1	9	20	14	8	7	9	18	21	5	8	9	13	9	9	15	13	4	4
9)	8	0	9	29	17	10	3	11	12	14	3	7	6	1	8	4	2	12	3	7
10)	2	2	11	13	10	9	5	10	10	13	3	5	6	4	9	8	4	9	1	7
11)	6	0	6	11	5	11	4	5	17	12	3	5	4	2	4	9	6	7	3	6
12)	6	2	4	23	11	7	0	15	21	13	2	7	7	5	4	2	2	7	3	11
13)	13	0	3	11	5	14	3	7	12	5	2	1	5	5	3	6	3	8	2	7
14)	17	1	13	11	5	11	4	15	9	9	3	7	8	6	11	8	8	11	3	5
15)	7	0	2	4	2	11	2	9	4	6	1	2	4	1	2	5	5	4	0	0
16)	9	0	4	8	4	13	1	7	10	9	3	2	3	7	19	7	5	10	0	5
17)	6	0	4	9	5	3	3	6	17	13	1	6	2	6	11	7	4	6	0	4
18)	1	0	4	9	9	7	0	7	14	13	1	5	2	1	17	3	2	10	2	2
19)	11	2	8	6	4	6	8	6	2	12	2	4	5	10	7	2	4	10	2	4
20)	3	6	8	6	8	6	4	8	5	15	2	6	7	9	7	13	4	12	0	5

⋮

C.10 *Amino acids frequency ranges (15 ranges for each amino acid)*

Item No. column represents Serial No. from 1 to 300. Data in the AA Frequency Range column represents single letter code of amino acid along with a frequency range. Frequency of each amino acid is divided into 15 partitions(ranges) resulting into 300 items for 20 amino acids. For example if frequency of amino acid Alanine (A) in a protein record is 22 it lies at Item No. 8.

Item No.	AA Frequency Range	Item No.	AA Frequency Range	Item No.	AA Frequency Range	Item No.	AA Frequency Range	Item No.	AA Frequency Range
1	<A:0..2>	61	<F:0..2>	121	<K:0..2>	181	<P:0..2>	241	<T:0..2>
2	<A:3..5>	62	<F:3..5>	122	<K:3..5>	182	<P:3..5>	242	<T:3..5>
3	<A:6..8>	63	<F:6..8>	123	<K:6..8>	183	<P:6..8>	243	<T:6..8>
4	<A:9..11>	64	<F:9..11>	124	<K:9..11>	184	<P:9..11>	244	<T:9..11>
5	<A:12..14>	65	<F:12..14>	125	<K:12..14>	185	<P:12..14>	245	<T:12..14>
6	<A:15..17>	66	<F:15..17>	126	<K:15..17>	186	<P:15..17>	246	<T:15..17>
7	<A:18..20>	67	<F:18..20>	127	<K:18..20>	187	<P:18..20>	247	<T:18..20>
8	<A:21..30>	68	<F:21..30>	128	<K:21..30>	188	<P:21..30>	248	<T:21..30>
9	<A:31..40>	69	<F:31..40>	129	<K:31..40>	189	<P:31..40>	249	<T:31..40>
10	<A:41..50>	70	<F:41..50>	130	<K:41..50>	190	<P:41..50>	250	<T:41..50>
11	<A:51..60>	71	<F:51..60>	131	<K:51..60>	191	<P:51..60>	251	<T:51..60>
12	<A:61..70>	72	<F:61..70>	132	<K:61..70>	192	<P:61..70>	252	<T:61..70>
13	<A:71..80>	73	<F:71..80>	133	<K:71..80>	193	<P:71..80>	253	<T:71..80>
14	<A:81..90>	74	<F:81..90>	134	<K:81..90>	194	<P:81..90>	254	<T:81..90>
15	<A:91..400>	75	<F:91..400>	135	<K:91..400>	195	<P:91..400>	255	<T:91..400>
16	<C:0..2>	76	<G:0..2>	136	<L:0..2>	196	<Q:0..2>	256	<V:0..2>
17	<C:3..5>	77	<G:3..5>	137	<L:3..5>	197	<Q:3..5>	257	<V:3..5>
18	<C:6..8>	78	<G:6..8>	138	<L:6..8>	198	<Q:6..8>	258	<V:6..8>
19	<C:9..11>	79	<G:9..11>	139	<L:9..11>	199	<Q:9..11>	259	<V:9..11>
20	<C:12..14>	80	<G:12..14>	140	<L:12..14>	200	<Q:12..14>	260	<V:12..14>
21	<C:15..17>	81	<G:15..17>	141	<L:15..17>	201	<Q:15..17>	261	<V:15..17>
22	<C:18..20>	82	<G:18..20>	142	<L:18..20>	202	<Q:18..20>	262	<V:18..20>
23	<C:21..30>	83	<G:21..30>	143	<L:21..30>	203	<Q:21..30>	263	<V:21..30>
24	<C:31..40>	84	<G:31..40>	144	<L:31..40>	204	<Q:31..40>	264	<V:31..40>
25	<C:41..50>	85	<G:41..50>	145	<L:41..50>	205	<Q:41..50>	265	<V:41..50>
26	<C:51..60>	86	<G:51..60>	146	<L:51..60>	206	<Q:51..60>	266	<V:51..60>
27	<C:61..70>	87	<G:61..70>	147	<L:61..70>	207	<Q:61..70>	267	<V:61..70>
28	<C:71..80>	88	<G:71..80>	148	<L:71..80>	208	<Q:71..80>	268	<V:71..80>
29	<C:81..90>	89	<G:81..90>	149	<L:81..90>	209	<Q:81..90>	269	<V:81..90>
30	<C:91..400>	90	<G:91..400>	150	<L:91..400>	210	<Q:91..400>	270	<V:91..400>

C.11 BDB_1 *at site* S_1

This Boolean data bank (BDB_1) is created using AAF_1 as shown in Appendix C.7 and amino acids frequency range table as shown in Appendix C.10. Each column heading represents an Item No. as shown in Appendix C.10. Each amino acid has 15 frequency partitions and for each amino acid, a boolean value '1' is put under that Item No. in which frequency of amino acid lies in a particular protein record otherwise a value '0' is considered. For example it clear from AAF_1 that the frequency of amino acid Alanine (A) in the 1^{st} protein record is 22 and this frequency lies at Item No. 8 in frequency range table as shown in Appendix C.10, so a boolean value '1' is put under Item No. 8 in the 1^{st} protein record in BDB_1.

S.No.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
1)	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
2)	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
3)	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
4)	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
5)	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
6)	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
7)	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
8)	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
9)	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
10)	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
11)	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
12)	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
13)	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
14)	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
15)	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
16)	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
17)	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
18)	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
19)	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
20)	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
21)	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
22)	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
23)	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
24)	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
25)	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
26)	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
27)	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
28)	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
29)	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
30)	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0

C.12 BDB_2 *at site* S_2

This Boolean data bank (BDB_2) is created using AAF_2 as shown in Appendix C.8 and amino acids frequency range table as shown in Appendix C.10. Each column heading represents an Item No. as shown in Appendix C.10. Each amino acid has 15 frequency partitions and for each amino acid, a boolean value '1' is put under that Item No. in which frequency of amino acid lies in a particular protein record otherwise a value '0' is considered. For example it clear from AAF_2 that the frequency of amino acid Alanine (A) in the 1^{st} protein record is 14 and this frequency lies at Item No. 5 in frequency range table as shown in Appendix C.10, so a boolean value '1' is put under Item No. 5 in the 1^{st} protein record in BDB_2.

S.No.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
1)	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
2)	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
3)	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
4)	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
5)	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
6)	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
7)	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
8)	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
9)	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
10)	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
11)	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
12)	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
13)	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
14)	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
15)	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
16)	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
17)	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
18)	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
19)	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
20)	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
21)	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
22)	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
23)	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
24)	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
25)	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
26)	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
27)	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
28)	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
29)	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
30)	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

C.13 BDB_3 *at site* S_3

This Boolean data bank (BDB_3) is created using AAF_3 as shown in Appendix C.9 and amino acids frequency range table as shown in Appendix C.10. Each column heading represents an Item No. as shown in Appendix C.10. Each amino acid has 15 frequency partitions and for each amino acid, a boolean value '1' is put under that Item No. in which frequency of amino acid lies in a particular protein record otherwise a value '0' is considered. For example it clear from AAF_3 that the frequency of amino acid Alanine (A) in the 1^{st} protein record is 9 and this frequency lies at Item No. 4 in frequency range table as shown in Appendix C.10, so a boolean value '1' is put under Item No. 5 in the 1^{st} protein record in BDB_3.

S.No.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
1)	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
2)	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
3)	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
4)	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
5)	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
6)	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
7)	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
8)	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
9)	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
10)	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
11)	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
12)	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
13)	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
14)	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
15)	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
16)	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
17)	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
18)	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
19)	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
20)	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
21)	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
22)	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
23)	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
24)	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
25)	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
26)	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
27)	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
28)	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
29)	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
30)	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0

C.14 IDB_1 *at site* S_1

This Itemset data bank (IDB_1) is created using BDB_1 as shown in Appendix C.11. Each record in this dataset consist of 20 Item Numbers, one for each of amino acids for which a boolean value '1' is stored in BDB_1.

```
 1)    8  16  32  47  63  79  91 107 121 138 151 168 181 198 212 227 245 260 271 286
 2)    6  16  35  49  63  79  93 106 124 140 152 167 181 197 213 226 243 259 271 287
 3)    5  16  33  48  63  80  92 108 122 140 152 166 183 197 213 228 243 260 271 287
 4)    5  16  34  50  63  78  92 107 121 140 152 166 182 197 216 228 242 258 271 288
 5)    4  16  32  50  63  78  93 107 122 141 152 167 184 197 213 227 243 257 271 287
 6)    8  16  33  47  62  80  93 107 124 138 151 168 181 197 211 228 241 258 271 287
 7)    7  16  35  47  63  79  91 108 126 140 152 169 181 198 212 228 243 260 271 287
 8)    8  16  33  47  64  81  91 106 124 139 153 168 182 197 211 229 242 258 272 286
 9)    5  16  33  50  63  78  94 109 125 139 152 167 182 197 212 228 244 259 271 287
10)    8  16  33  48  62  83  92 108 124 139 152 167 182 197 212 230 241 259 271 287
11)    6  16  33  50  63  79  95 109 127 142 151 166 182 197 212 228 242 258 271 287
12)    8  16  33  47  66  79  91 107 124 139 152 169 183 196 212 230 241 259 271 286
13)    6  16  34  47  65  79  92 109 124 138 152 167 182 197 212 229 244 259 271 286
14)    8  16  33  50  63  78  92 109 125 140 151 168 182 197 211 229 243 261 272 286
15)    6  17  33  49  63  77  94 107 124 142 151 169 182 197 212 228 243 259 271 287
16)    5  16  34  47  62  78  92 109 125 140 152 167 183 196 212 230 242 259 271 287
17)    5  16  33  49  64  78  93 108 126 140 151 167 181 199 212 228 243 260 271 287
18)    6  16  35  47  63  79  93 109 124 140 152 168 182 197 212 229 243 259 271 288
19)    3  16  36  47  64  77  93 109 125 139 152 166 182 198 213 228 244 260 272 287
20)    3  16  33  49  64  77  91 111 127 140 151 168 183 198 211 230 242 258 271 287
                                        ⋮
```

C.15 IDB_2 *at site* S_2

This Itemset data bank (IDB_2) is created using BDB_2 as shown in Appendix C.12. Each record in this dataset consist of 20 Item Numbers, one for each of amino acids for which a boolean value '1' is stored in BDB_2.

```
 1)    5  16  32  50  63  80  92 109 124 140 151 170 184 198 214 227 245 261 271 287
 2)    2  16  35  48  63  80  91 108 123 137 153 168 182 199 214 228 244 259 272 289
 3)    6  16  34  49  62  77  92 108 123 138 152 167 184 196 214 227 243 259 271 287
 4)    3  16  33  49  63  79  92 107 122 138 152 167 183 196 216 228 243 259 272 286
 5)    4  16  34  48  62  82  93 109 123 140 153 170 183 199 213 229 244 260 272 288
 6)    4  17  34  48  61  80  92 109 124 138 153 168 183 196 213 229 243 258 271 290
 7)    6  16  31  49  61  78  92 106 122 138 151 166 183 197 213 226 242 258 271 286
 8)    3  16  31  47  62  77  92 107 122 138 151 167 182 197 212 229 242 258 271 287
 9)    3  16  32  46  63  77  91 108 122 137 151 170 182 197 212 227 242 257 271 287
10)    2  16  33  48  61  78  91 108 123 137 152 168 181 196 215 228 241 260 271 286
11)    1  17  33  47  61  77  91 108 122 139 151 168 182 197 214 228 241 259 271 286
12)    3  16  32  47  62  78  92 106 121 139 151 167 182 198 213 228 242 259 271 286
13)    4  16  32  48  62  77  92 108 124 137 152 167 182 197 213 226 241 258 271 287
14)    4  16  34  52  63  79  93 108 124 143 152 167 186 197 216 229 242 263 271 288
15)    5  16  34  49  63  80  92 109 124 140 152 167 185 198 214 228 245 263 271 287
16)    7  16  35  51  64  80  92 109 125 140 152 169 185 197 216 229 243 262 271 288
17)    3  17  33  51  64  79  91 107 123 139 151 167 183 199 216 228 242 258 272 287
18)    4  16  31  48  61  77  91 107 123 138 152 166 182 196 211 226 242 257 271 286
19)    2  16  34  49  62  77  92 108 124 140 151 166 182 199 214 229 242 257 272 286
20)    4  16  32  49  62  77  92 107 125 138 152 168 181 197 212 229 241 258 272 287
                                        ⋮
```

C.16 IDB_3 *at site* S_3

This Itemset data bank (IDB_3) is created using BDB_3 as shown in Appendix C.13. Each record in this dataset consist of 20 Item Numbers, one for each of amino acids for which a boolean value '1' is stored in BDB_3.

```
 1)    4  16  33  49  61  76  92 107 123 141 152 167 183 199 212 229 242 259 271 287
 2)    3  16  34  47  64  77  92 109 123 139 152 167 184 199 212 227 244 256 272 287
 3)    3  16  33  48  62  79  92 107 122 139 151 167 182 196 214 228 243 258 271 289
 4)    3  16  32  48  63  78  91 107 124 139 152 167 182 197 212 229 242 258 271 287
 5)    2  18  32  46  63  77  92 108 126 136 151 168 181 196 212 230 242 257 271 287
 6)    2  16  31  47  61  78  92 108 124 137 151 167 182 197 212 227 242 257 271 287
 7)    2  16  33  49  62  78  91 110 124 139 153 169 181 196 212 227 242 257 271 287
 8)    5  16  34  52  65  78  93 109 127 143 152 168 184 200 214 229 246 260 272 287
 9)    3  16  34  53  66  79  92 109 125 140 152 168 183 196 213 227 241 260 272 288
10)    1  16  34  50  64  79  92 109 124 140 152 167 183 197 214 228 242 259 271 288
11)    3  16  33  49  62  79  92 107 126 140 152 167 182 196 212 229 243 258 272 288
12)    3  16  32  53  64  78  91 111 128 140 151 168 183 197 212 226 241 258 272 289
13)    5  16  32  49  62  80  92 108 125 137 151 166 182 197 212 228 242 258 271 288
14)    6  16  35  49  62  79  92 111 124 139 152 168 183 198 214 228 243 259 272 287
15)    3  16  31  47  61  79  91 109 122 138 151 166 182 196 211 227 242 257 271 286
16)    4  16  32  48  62  80  91 108 124 139 152 166 182 198 217 228 242 259 271 287
17)    3  16  32  49  62  77  92 108 126 140 151 168 181 198 214 228 242 258 271 287
18)    1  16  32  49  64  78  91 108 125 140 151 167 181 196 216 227 241 259 271 286
19)    4  16  33  48  62  78  93 108 121 140 151 167 182 199 213 226 242 259 271 287
20)    2  18  33  48  63  78  92 108 122 141 151 168 183 199 213 230 242 260 271 287
                                    ⋮
```

Appendix D - Resultant Knowledge of AeQARM-AAPDB System

D.1 $L^{FI}_{k(1)}$ *and* $L^{FISC}_{k(1)}$ *at site* S_1

List of frequent k-itemset, i.e., $L^{FI}_{k(1)}$ is represented by column **L** and column **SC** shows the support count of corresponding frequent k-itemset ($L^{FISC}_{k(1)}$) at site S_1. These frequent itemsets and their support counts are obtained by processing the Itemset Data Bank (IDB_1) as shown in Appendix C.14.

Sr. No.	1-Itemsets		2-Itemsets		3-Itemsets		4-Itemsets	
	L	SC	L	SC	L	SC	L	SC
1	[2]	713	[16, 32]	852	[16, 32, 271]	735	[16, 91, 151, 271]	869
2	[3]	765	[16, 33]	731	[16, 61, 271]	736		
3	[16]	2508	[16, 48]	769	[16, 62, 271]	757		
4	[32]	1015	[16, 61]	838	[16, 91, 151]	998		
5	[33]	923	[16, 62]	933	[16, 91, 271]	1252		
6	[48]	955	[16, 91]	1473	[16, 91, 286]	670		
7	[49]	695	[16, 92]	784	[16, 107, 271]	691		
8	[61]	948	[16, 107]	831	[16, 151, 271]	1239		
9	[62]	1172	[16, 108]	684	[16, 167, 271]	778		
10	[77]	796	[16, 123]	693	[16, 182, 271]	769		
11	[78]	849	[16, 151]	1492	[16, 197, 271]	800		
12	[91]	1769	[16, 152]	811	[16, 212, 271]	728		
13	[92]	1073	[16, 166]	690	[16, 227, 271]	680		
14	[107]	1010	[16, 167]	918	[16, 242, 271]	724		
15	[108]	870	[16, 181]	674	[16, 271, 286]	826		
16	[122]	767	[16, 182]	917	[16, 271, 287]	750		
17	[123]	874	[16, 197]	980	[91, 151, 271]	999		
18	[138]	708	[16, 212]	859	[91, 271, 286]	672		
19	[139]	682	[16, 213]	679				
20	[151]	1813	[16, 227]	766				
21	[152]	1112	[16, 242]	836				
22	[166]	781	[16, 271]	1961				
23	[167]	1145	[16, 286]	952				
24	[168]	747	[16, 287]	932				
25	[181]	733	[32, 91]	670				

D.2 $L_{k(2)}^{FI}$ *and* $L_{k(2)}^{FISC}$ *at site* S_2

List of frequent k-itemset, i.e., $L_{k(2)}^{FI}$ is represented by column **L** and column **SC** shows the support count of corresponding frequent k-itemset ($L_{k(2)}^{FISC}$) at site S_2. These frequent itemsets and their support counts are obtained by processing the Itemset Data Bank (IDB_2) as shown in Appendix C.15.

Sr.	1-Itemsets		2-Itemsets	
No.	L	SC	L	SC
1	[8]	753	[16, 92]	790
2	[16]	1960	[16, 151]	679
3	[17]	934	[16, 152]	760
4	[34]	666	[16, 271]	1260
5	[62]	737	[16, 287]	677
6	[63]	803	[91, 271]	669
7	[83]	719	[92, 271]	767
8	[91]	865	[151, 271]	652
9	[92]	1188	[152, 271]	732
10	[93]	684	[197, 271]	665
11	[143]	991	[271, 287]	723
12	[151]	936		
13	[152]	1213		
14	[153]	711		
15	[167]	777		

D.3 $L_{k(3)}^{FI}$ *and* $L_{k(3)}^{FISC}$ *at site* S_3

List of frequent k-itemset, i.e., $L_{k(3)}^{FI}$ is represented by column **L** and column **SC** shows the support count of corresponding frequent k-itemset ($L_{k(3)}^{FISC}$) at site S_3. These frequent itemsets and their support counts are obtained by processing the Itemset Data Bank (IDB_3) as shown in Appendix C.16.

Sr.	1-Itemsets		2-Itemsets		3-Itemsets	
No.	L	SC	L	SC	L	SC
1	[16]	1960	[16, 62]	675	[16, 91, 271]	776
2	[17]	684	[16, 91]	924	[16, 151, 271]	751
3	[32]	740	[16, 92]	705	[91, 151, 271]	710
4	[33]	784	[16, 151]	910		
5	[48]	672	[16, 152]	765		
6	[61]	623	[16, 167]	712		
7	[62]	982	[16, 182]	695		
8	[63]	732	[16, 197]	734		
9	[78]	721	[16, 271]	1433		
10	[91]	1329	[16, 287]	731		
11	[92]	1061	[32, 271]	635		
12	[107]	665	[62, 271]	780		
13	[108]	751	[91, 151]	811		
14	[122]	696	[91, 271]	1108		
15	[123]	726	[92, 271]	717		
16	[151]	1401	[151, 271]	1138		
17	[152]	1110	[152, 271]	736		
18	[166]	687	[167, 271]	816		
19	[167]	1033	[182, 271]	813		
20	[168]	638	[196, 271]	633		

D.4 L_1^{LSAR} *for Item numbers at site* S_1

Column **L** represents frequent k-itemset and column **AR(support,confidence)** shows the list of locally strong association rules, i.e., L_1^{LSAR} at site S_1. Each strong rule has its associated support and confidence factor. The minimum threshold support is taken as 20% and minimum threshold confidence as 50% for generating the strong rules by making use of the data as shown in Appendix D.1.

Sr. No.	Strong ARs for Frequent 4-itemsets		Strong ARs for Frequent 3-itemsets		Strong ARs for Frequent 2-itemsets	
	L	AR(support,confidence)	L	AR(support,confidence)	L	AR(support,confidence)
1	[16, 91, 151, 271]	[16, 91] => [151, 271] (26%, 58%)	[16, 32, 271]	[32] => [16, 271] (21%, 72%)	[16, 32]	[32] => [16] (25%, 83%)
2		[16, 151] => [91, 271] (26%, 58%)		[16, 32] => [271] (21%, 86%)	[16, 33]	[33] => [16] (21%, 79%)
3		[91, 151] => [16, 271] (26%, 74%)		[32, 271] => [16] (21%, 84%)	[16, 48]	[48] => [16] (23%, 80%)
4		[91, 271] => [16, 151] (26%, 59%)	[16, 61, 271]	[61] => [16, 271] (22%, 77%)	[16, 61]	[61] => [16] (25%, 88%)
5		[151, 271] => [16, 91] (26%, 59%)		[16, 61] => [271] (22%, 87%)	[16, 62]	[62] => [16] (27%, 79%)
6		[16, 91, 151] => [271] (26%, 87%)		[61, 271] => [16] (22%, 89%)	[16, 91]	[16] => [91] (44%, 58%)
7		[16, 91, 271] => [151] (26%, 69%)	[16, 62, 271]	[62] => [16, 271] (22%, 64%)		[91] => [16] (44%, 83%)
8		[16, 151, 271] => [91] (26%, 70%)		[16, 62] => [271] (22%, 81%)	[16, 92]	[92] => [16] (23%, 73%)
9		[91, 151, 271] => [16] (26%, 86%)		[62, 271] => [16] (22%, 81%)	[16, 107]	[107] => [16] (24%, 82%)
10			[16, 91, 151]	[91] => [16, 151] (29%, 56%)	[16, 108]	[108] => [16] (20%, 78%)
11				[151] => [16, 91] (29%, 55%)	[16, 123]	[123] => [16] (20%, 79%)
12				[16, 91] => [151] (29%, 67%)	[16, 151]	[16] => [151] (44%, 59%)
13				[16, 151] => [91] (29%, 66%)		[151] => [16] (44%, 82%)
14				[91, 151] => [16] (29%, 85%)	[16, 152]	[152] => [16] (24%, 72%)
15			[16, 91, 271]	[91] => [16, 271] (37%, 70%)	[16, 166]	[166] => [16] (20%, 88%)
16				[271] => [16, 91] (37%, 50%)	[16, 167]	[167] => [16] (27%, 80%)
17				[16, 91] => [271] (37%, 84%)	[16, 181]	[181] => [16] (20%, 91%)
18				[16, 271] => [91] (37%, 63%)	[16, 182]	[182] => [16] (27%, 79%)
19				[91, 271] => [16] (37%, 85%)	[16, 197]	[197] => [16] (29%, 80%)
20			[16, 91, 286]	[286] => [16, 91] (20%, 60%)	[16, 212]	[212] => [16] (25%, 81%)
21				[16, 286] => [91] (20%, 70%)	[16, 213]	[213] => [16] (20%, 76%)
22				[91, 286] => [16] (20%, 88%)	[16, 227]	[227] => [16] (22%, 86%)
23			[16, 107, 271]	[107] => [16, 271] (20%, 68%)	[16, 242]	[242] => [16] (25%, 82%)
24				[16, 107] => [271] (20%, 83%)	[16, 271]	[16] => [271] (58%, 78%)
25				[107, 271] => [16] (20%, 83%)		[271] => [16] (58%, 79%)

⋮

D.5 L_1^{LSAR} *for corresponding Amino acid frequency ranges at site* S_1

Replace Item No. in Appendix D.4 data with its corresponding Amino acid frequency range as shown in table Appendix C.10.

Sr. No.	Strong ARs for Frequent 4-itemsets		Strong ARs for Frequent 3-itemsets	
	L	AR(support,confidence)	L	AR(support,confidence)
1	{ <C:0..2> <H:0..2> <M:0..2> <W:0..2> }	{ <C:0..2> <H:0..2> } => { <M:0..2> <W:0..2> } (26%, 58%)	{ <C:0..2> <D:3..5> <W:0..2> }	{ <D:3..5> } => { <C:0..2> <W:0..2> } (21%, 72%)
2		{ <C:0..2> <M:0..2> } => { <H:0..2> <W:0..2> } (26%, 58%)		{ <C:0..2> <D:3..5> } => { <W:0..2> } (21%, 86%)
3		{ <H:0..2> <M:0..2> } => { <C:0..2> <W:0..2> } (26%, 74%)		{ <D:3..5> <W:0..2> } => { <C:0..2> } (21%, 84%)
4		{ <H:0..2> <W:0..2> } => { <C:0..2> <M:0..2> } (26%, 59%)	{ <C:0..2> <F:0..2> <W:0..2> }	{ <F:0..2> } => { <C:0..2> <W:0..2> } (22%, 77%)
5		{ <M:0..2> <W:0..2> } => { <C:0..2> <H:0..2> } (26%, 59%)		{ <C:0..2> <F:0..2> } => { <W:0..2> } (22%, 87%)
6		{ <C:0..2> <H:0..2> <M:0..2> } => { <W:0..2> } (26%, 87%)		{ <F:0..2> <W:0..2> } => { <C:0..2> } (22%, 89%)
7		{ <C:0..2> <H:0..2> <W:0..2> } => { <M:0..2> } (26%, 69%)	{ <C:0..2> <F:3..5> <W:0..2> }	{ <F:3..5> } => { <C:0..2> <W:0..2> } (22%, 64%)
8		{ <C:0..2> <M:0..2> <W:0..2> } => { <H:0..2> } (26%, 70%)		{ <C:0..2> <F:3..5> } => { <W:0..2> } (22%, 81%)
9		{ <H:0..2> <M:0..2> <W:0..2> } => { <C:0..2> } (26%, 86%)		{ <F:3..5> <W:0..2> } => { <C:0..2> } (22%, 81%)
10			{ <C:0..2> <H:0..2> <M:0..2> }	{ <H:0..2> } => { <C:0..2> <M:0..2> } (29%, 56%)
11				{ <M:0..2> } => { <C:0..2> <H:0..2> } (29%, 55%)
12				{ <C:0..2> <H:0..2> } => { <M:0..2> } (29%, 67%)
13				{ <C:0..2> <M:0..2> } => { <H:0..2> } (29%, 66%)
14				{ <H:0..2> <M:0..2> } => { <C:0..2> } (29%, 85%)
15			{ <C:0..2> <H:0..2> <W:0..2> }	{ <H:0..2> } => { <C:0..2> <W:0..2> } (37%, 70%)
16				{ <W:0..2> } => { <C:0..2> <H:0..2> } (37%, 50%)
17				{ <C:0..2> <H:0..2> } => { <W:0..2> } (37%, 84%)
18				{ <C:0..2> <W:0..2> } => { <H:0..2> } (37%, 63%)
19				{ <H:0..2> <W:0..2> } => { <C:0..2> } (37%, 85%)
20			{ <C:0..2> <H:0..2> <Y:0..2> }	{ <Y:0..2> } => { <C:0..2> <H:0..2> } (20%, 60%)
21				{ <C:0..2> <Y:0..2> } => { <H:0..2> } (20%, 70%)
22				{ <H:0..2> <Y:0..2> } => { <C:0..2> } (20%, 88%)

⋮

D.6 L_2^{LSAR} *for Item numbers at site* S_2

Column **L** represents frequent k-itemset and column **AR(support,confidence)** shows the list of locally strong association rules, i.e., L_2^{LSAR} at site S_2. Each strong rule has its associated support and confidence factor. The minimum threshold support is taken as 20% and minimum threshold confidence as 50% for generating the strong rules by making use of the data as shown in Appendix D.2.

Sr. No.	Strong ARs for Frequent 2-itemsets	
	L	AR(support,confidence)
1	[16, 92]	[92] => [16] (24%, 66%)
2	[16, 151]	[151] => [16] (20%, 72%)
3	[16, 152]	[152] => [16] (23%, 62%)
4	[16, 271]	[16] => [271] (38%, 64%)
5		[271] => [16] (38%, 66%)
6	[16, 287]	[287] => [16] (20%, 66%)
7	[91, 271]	[91] => [271] (20%, 77%)
8	[92, 271]	[92] => [271] (23%, 64%)
9	[151, 271]	[151] => [271] (20%, 69%)
10	[152, 271]	[152] => [271] (22%, 60%)
11	[197, 271]	[197] => [271] (20%, 73%)
12	[271, 287]	[287] => [271] (22%, 71%)

D.7 L_2^{LSAR} *for corresponding Amino acid frequency ranges at site* S_2

Replace Item No. in Appendix D.6 data with its corresponding Amino acid frequency range as shown in table Appendix C.10.

Sr. No.	Strong ARs for Frequent 2-itemsets	
	L	AR(support,confidence)
1	{ <C:0..2> <H:3..5> }	{ <H:3..5> } => { <C:0..2> } (24%, 66%)
2	{ <C:0..2> <M:0..2> }	{ <M:0..2> } => { <C:0..2> } (20%, 72%)
3	{ <C:0..2> <M:3..5> }	{ <M:3..5> } => { <C:0..2> } (23%, 62%)
4	{ <C:0..2> <W:0..2> }	{ <C:0..2> } => { <W:0..2> } (38%, 64%)
5		{ <W:0..2> } => { <C:0..2> } (38%, 66%)
6	{ <C:0..2> <Y:3..5> }	{ <Y:3..5> } => { <C:0..2> } (20%, 66%)
7	{ <H:0..2> <W:0..2> }	{ <H:0..2> } => { <W:0..2> } (20%, 77%)
8	{ <H:3..5> <W:0..2> }	{ <H:3..5> } => { <W:0..2> } (23%, 64%)
9	{ <M:0..2> <W:0..2> }	{ <M:0..2> } => { <W:0..2> } (20%, 69%)
10	{ <M:3..5> <W:0..2> }	{ <M:3..5> } => { <W:0..2> } (22%, 60%)
11	{ <Q:3..5> <W:0..2> }	{ <Q:3..5> } => { <W:0..2> } (20%, 73%)
12	{ <W:0..2> <Y:3..5> }	{ <Y:3..5> } => { <W:0..2> } (22%, 71%)

D.8 L_3^{LSAR} *for Item numbers at site* S_3

Column **L** represents frequent k-itemset and column **AR(support,confidence)** shows the list of locally strong association rules, i.e., L_3^{LSAR} at site S_3. Each strong rule has its associated support and confidence factor. The minimum threshold support is taken as 20% and minimum threshold confidence as 50% for generating the strong rules by making use of the data as shown in Appendix D.3.

Sr. No.	Strong ARs for Frequent 3-itemsets		Strong ARs for Frequent 2-itemsets	
	L	AR(support,confidence)	L	AR(support,confidence)
1	[16, 91, 271]	[91] => [16, 271] (25%, 58%)	[16, 62]	[62] => [16] (22%, 68%)
2		[16, 91] => [271] (25%, 83%)	[16, 91]	[91] => [16] (30%, 69%)
3		[16, 271] => [91] (25%, 54%)	[16, 92]	[92] => [16] (23%, 66%)
4		[91, 271] => [16] (25%, 70%)	[16, 151]	[151] => [16] (29%, 64%)
5	[16, 151, 271]	[151] => [16, 271] (24%, 53%)	[16, 152]	[152] => [16] (25%, 68%)
6		[16, 151] => [271] (24%, 82%)	[16, 167]	[167] => [16] (23%, 68%)
7		[16, 271] => [151] (24%, 52%)	[16, 182]	[182] => [16] (22%, 69%)
8		[151, 271] => [16] (24%, 65%)	[16, 197]	[197] => [16] (24%, 70%)
9	[91, 151, 271]	[91] => [151, 271] (23%, 53%)	[16, 271]	[16] => [271] (47%, 73%)
10		[151] => [91, 271] (23%, 50%)		[271] => [16] (47%, 67%)
11		[91, 151] => [271] (23%, 87%)	[16, 287]	[287] => [16] (24%, 68%)
12		[91, 271] => [151] (23%, 64%)	[32, 271]	[32] => [271] (20%, 85%)
13		[151, 271] => [91] (23%, 62%)	[62, 271]	[62] => [271] (25%, 79%)
14			[91, 151]	[91] => [151] (26%, 61%)
15				[151] => [91] (26%, 57%)
16			[91, 271]	[91] => [271] (36%, 83%)
17				[271] => [91] (36%, 52%)
18			[92, 271]	[92] => [271] (23%, 67%)
19			[151, 271]	[151] => [271] (37%, 81%)
20				[271] => [151] (37%, 53%)

⋮

D.9 L_3^{LSAR} *for corresponding Amino acid frequency ranges at site* S_3

Replace Item No. in Appendix D.8 data with its corresponding Amino acid frequency range as shown in table Appendix C.10.

Sr. No.	Strong ARs for Frequent 3-itemsets		Strong ARs for Frequent 2-itemsets	
	L	AR(support,confidence)	L	AR(support,confidence)
1	{ <C:0..2> <H:0..2> <W:0..2> }	{ <H:0..2> } => { <C:0..2> <W:0..2> } (25%, 58%)	{ <C:0..2> <F:3..5> }	{ <F:3..5> } => { <C:0..2> } (22%, 68%)
2		{ <C:0..2> <H:0..2> } => { <W:0..2> } (25%, 83%)	{ <C:0..2> <H:0..2> }	{ <H:0..2> } => { <C:0..2> } (30%, 69%)
3		{ <C:0..2> <W:0..2> } => { <H:0..2> } (25%, 54%)	{ <C:0..2> <H:3..5> }	{ <H:3..5> } => { <C:0..2> } (23%, 66%)
4		{ <H:0..2> <W:0..2> } => { <C:0..2> } (25%, 70%)	{ <C:0..2> <M:0..2> }	{ <M:0..2> } => { <C:0..2> } (29%, 64%)
5	{ <C:0..2> <M:0..2> <W:0..2> }	{ <M:0..2> } => { <C:0..2> <W:0..2> } (24%, 53%)	{ <C:0..2> <M:3..5> }	{ <M:3..5> } => { <C:0..2> } (25%, 68%)
6		{ <C:0..2> <M:0..2> } => { <W:0..2> } (24%, 82%)	{ <C:0..2> <N:3..5> }	{ <N:3..5> } => { <C:0..2> } (23%, 68%)
7		{ <C:0..2> <W:0..2> } => { <M:0..2> } (24%, 52%)	{ <C:0..2> <P:3..5> }	{ <P:3..5> } => { <C:0..2> } (22%, 69%)
8		{ <M:0..2> <W:0..2> } => { <C:0..2> } (24%, 65%)	{ <C:0..2> <Q:3..5> }	{ <Q:3..5> } => { <C:0..2> } (24%, 70%)
9	{ <H:0..2> <M:0..2> <W:0..2> }	{ <H:0..2> } => { <M:0..2> <W:0..2> } (23%, 53%)	{ <C:0..2> <W:0..2> }	{ <C:0..2> } => { <W:0..2> } (47%, 73%)
10		{ <M:0..2> } => { <H:0..2> <W:0..2> } (23%, 50%)		{ <W:0..2> } => { <C:0..2> } (47%, 67%)
11		{ <H:0..2> <M:0..2> } => { <W:0..2> } (23%, 87%)	{ <C:0..2> <Y:3..5> }	{ <Y:3..5> } => { <C:0..2> } (24%, 68%)
12		{ <H:0..2> <W:0..2> } => { <M:0..2> } (23%, 64%)	{ <D:3..5> <W:0..2> }	{ <D:3..5> } => { <W:0..2> } (20%, 85%)
13		{ <M:0..2> <W:0..2> } => { <H:0..2> } (23%, 62%)	{ <F:3..5> <W:0..2> }	{ <F:3..5> } => { <W:0..2> } (25%, 79%)
14			{ <H:0..2> <M:0..2> }	{ <H:0..2> } => { <M:0..2> } (26%, 61%)
15				{ <M:0..2> } => { <H:0..2> } (26%, 57%)
16			{ <H:0..2> <W:0..2> }	{ <H:0..2> } => { <W:0..2> } (36%, 83%)
17				{ <W:0..2> } => { <H:0..2> } (36%, 52%)
18			{ <H:3..5> <W:0..2> }	{ <H:3..5> } => { <W:0..2> } (23%, 67%)
19			{ <M:0..2> <W:0..2> }	{ <M:0..2> } => { <W:0..2> } (37%, 81%)
20				{ <W:0..2> } => { <M:0..2> } (37%, 53%)

⋮

LIST OF PUBLICATIONS

[1] Gurpreet S. Bhamra, Anil K. Verma, and Ram B. Patel. An Encounter with Strong Association Rules. In *Proceedings of IEEE International Advanced Computing Conference (IACC-2010)*, pages 342–346. Thapar University, Patiala, Punjab, India, IEEE, Feb 19-20 2010.

[2] Gurpreet S. Bhamra, Anil K. Verma, and Ram B. Patel. Agent Enriched Distributed Association Rules Mining: A Review. In Longbing Cao, Ana L. C. Bazzan, Andreas L. Symeonidis, Vladimir I. Gorodetsky, Gerhard Weiss, and Philip S. Yu, editors, *Agents and Data Mining Interaction*, volume 7103 of *Lecture Notes in Computer Science*, pages 30–45. Springer Berlin - Heidelberg, 2012.

[3] Gurpreet S. Bhamra, Anil K. Verma, and Ram B. Patel. Agent Technology in Bioinformatics: A Review. *International Journal of Next-Generation Computing*, 5(2):176–189, July 2014.

[4] Gurpreet S. Bhamra, Anil K. Verma, and Ram B. Patel. A Serial Computing Model of agent enabled Mining of Globally Strong Association Rules. *International Journal on Computational Science & Applications (IJCSA)*, 5(3):77–104, June 2015.

[5] Gurpreet S. Bhamra, Anil K. Verma, and Ram B. Patel. Agent Based Frameworks for Distributed Association Rule Mining: An Analysis. *International Journal in Foundations of Computer Science & Technology (IJFCST)*, 5(1):11–22, January 2015.

[6] Gurpreet S. Bhamra, Anil K. Verma, and Ram B. Patel. Agent enabled Mining of Distributed Protein Data Banks. *International Journal in Foundations of Computer Science & Technology (IJFCST)*, 5(3):25–45, May 2015.

[7] Gurpreet Singh Bhamra, Ram Bahadur Patel, and Anil Kumar Verma. A Parallel Computing Model of agent enabled Mining of Globally Strong Association Rules. *IEEE Transactions on Knowledge and Data Engineering (TKDE).* (SCI-E indexed, Communicated).

[8] Gurpreet Singh Bhamra, Ram Bahadur Patel, and Anil Kumar Verma. Agent enriched Quantitative Association Rules Mining of Amino Acids from distributed Protein Data Banks. *Chiang Mai Journal of Science (CMJS).* (SCI-E indexed, Communicated).

REFERENCES

[1] Cristian Aflori and Florin Leon. Efficient Distributed Data Mining using Intelligent Agents . In *Proceedings of the 8th International Symposium on Automatic Control and Computer Science* , pages 1–6, 2004.

[2] AgentLink. AgentLink: European Commission's 6th Framework Programme. http://www.agentlink.org, 2004.

[3] R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In *Proceedings of the ACM-SIGMOD International Conference of Management of Data*, pages 207–216, 1993.

[4] R. Agrawal and J. C. Shafer. Parallel mining of association rules. *IEEE Transaction on Knowledge and Data Engineering*, 8(6):962–969, 1996.

[5] Rakesh Agrawal and Ramakrishnan Srikant. Fast Algorithms for Mining Association Rules in Large Databases. In *Proceedings of the 20th International Conference on Very Large Data Bases(VLDB'94)*, pages 487–499. Morgan Kaufmann Publishers Inc., Sept. 12-15 1994.

[6] Kamal Ali Albashiri. *An investigation into the issues of Multi-Agent Data Mining.* PhD thesis, The University of Liverpool, Ashton Building, Ashton Street, Liverpool L69 3BX, United Kingdom, February 2010.

[7] Kamal Ali Albashiri and Frans Coenen. Agent-Enriched Data Mining Using an Extendable Framework. In Longbing Cao, Vladimir Gorodetsky, Jiming Liu, Gerhard Weiss, and Philip S. Yu, editors, *Agents and Data Mining Interaction*, volume 5680 of *Lecture Notes in Computer Science*, pages 53–68. Springer Berlin - Heidelberg, 2009a.

[8] Kamal Ali Albashiri, Frans Coenen, and Paul Leng. EMADS: An extendible multi-agent data miner. *Knowledge-Based Systems*, 22(7):523–528, October 2009b.

[9] Juan M. Alberola, Jose M. Such, Ana Garcia-Fornes, Agustin Espinosa, and Vicent Botti. A performance evaluation of three multiagent platforms. *Artificial Intelligence Review*, 34(2):145–176, 2010.

[10] Ezendu Ifeanyi Ariwa, Mohamed B. Senousy, and Mohamed M. Medhat. Informatization and E-Business Model Application for Distributed Data Mining Using Mobile Agents . In *Proceedings of the IADIS International Conference WWW/Internet 2003,ICWI 2003*, pages 85–92, November 5-8 2003.

[11] AstralSCOPv1.63. Astral SCOP version 1.63 genetic domain sequence subsets with less than 40% identity to each other. http://scop.berkeley.edu/astral/ver=1.63, June 2003.

[12] AstralSCOPv1.75. Astral SCOP version 1.75 genetic domain sequence subsets with less than 40% identity to each other. http://scop.berkeley.edu/astral/ver=1.75, June 2009.

[13] Amir F. Atiya and Yaser S. Abu-Mostafa. Introduction to financial forecasting. *Applied Intelligence*, 6(3):205–213, July 1996.

[14] T. Attwood and D. Parry-Smith. *Introduction to Bioinformatics*. Prentice Hall, First edition, March 1999.

[15] Sung Wook Baik, Jerzy Bala, and Ju Sang Cho. Agent Based Distributed Data Mining. In Kim-Meow Liew, Hong Shen, Simon See, Wentong Cai, Pingzhi Fan, and Susumu Horiguchi, editors, *Parallel and Distributed Computing: Applications and Technologies*, volume 3320 of *Lecture Notes in Computer Science*, pages 42–45. Springer Berlin - Heidelberg, 2005.

[16] S. Bailey, R. Grossman, H. Sivakumar, and A. Turinsky. Papyrus: a system for data mining over local and wide area clusters and super-clusters . In *Proceedings of the ACM/IEEE conference on Supercomputing (SC'99)*, page 63. ACM New York, NY, USA, 1999.

[17] E. Bartocci, D. Cacciagrano, N. Cannata, F. Corradini, E. Merelli, L. Milanesi, and P. Romano. An agent-based multilayer architecture for bioinformatics grids. *IEEE Transactions on NanoBioscience*, 6(2):142–148, 2007.

[18] E. Bartocci, F. Corradini, and E. Merelli. Building a multiagent system from a user workfow specifcation. In Flavio De Paoli, Antonella Di Stefano, Andrea Omicini, and Corrado Santoro, editors, *Proceedings of the 7th WOA 2006 Workshop, From Objects to Agents (Dagli Oggetti Agli Agenti)*, pages 96–103, Catania, Italy, September 2006.

[19] Ana L.C. Bazzan. Agents and data mining in bioinformatics: Joining data gathering and automatic annotation with classification and distributed clustering. In L. Cao, V. Gorodetsky, J. Liu, G. Weiss, and P. S. Yu, editors, *Agents and Data Mining Interaction*, volume 5680 of *Lecture Notes in Computer Science*, pages 3–20. Springer Berlin - Heidelberg, 2009. See also HermesV2 http://hermes.cs.unicam.it.

[20] Fabio Luigi Bellifemine, Giovanni Caire, and Dominic Greenwood. *Developing Multi-Agent Systems with JADE*. Wiley Series in Agent Technology. John Wiley & Sons, New York, February 2007.

[21] D. Bonura, L. Mariani, and E. Merelli. Designing modular agent systems. In *Proceedings of NET.Object DAYS*, pages 245–263, Erfurt, September 2003.

[22] L. Bortolussi, A. Dovier, and F. Fogolari. Multi-agent simulation of protein folding. In *Proceedings of the 1st AAMAS International Workshop on Multi-Agent Systems*

*for Medicine, Computational Biology, and Bioinformatics (MAS*BIOMED'05)*, pages 91–106, Utrecht,Netherlands, July 25 2005.

[23] Jeffrey M. Bradshaw. *Software agents*. MIT Press Cambridge, MA, USA, 1997.

[24] K. Bryson, M. Luck, M. Joy, and D.T. Jones. Applying agents to bioinformatics in geneweaver. In Matthias Klusch and Larry Kerschberg, editors, *Cooperative Information Agents IV- The Future of Information Agents in Cyberspace*, volume 1860 of *Lecture Notes in Artificial Intelligence*, pages 60–71. Springer-Verlag, 2000.

[25] Michael Byrd and Conny Franke. The State of Distributed Data Mining. Technical Report ECS265, UC Davis, Davis CA 95616, USA, 2007.

[26] L. Cao, G. Weiss, and P. S. Yu. A brief introduction to agent mining. *Autonomous Agent and Multi-Agent System*, 25:419–424, 2012.

[27] R. C. Cardoso, C. Diego, M. Bordini, and H. Rafael. Agent Programming and Multi-agent Systems project at Benchmarking Agent Programming Languages (APLs). http://agentprogramming.com/, 2013.

[28] Zineb Chaouch and Mohammed Tamali. A Mobile Agent-Based Technique for Medical Monitoring (Supports of Patients with Diabetes). *International Journal of Computational Models and Algorithms in Medicine (IJCMAM)*, 4(1):17–32, March 2014.

[29] J. Chattratichat, J. Darlington, Y. Guo, S. Hedvall, M. Kohler, and J. Syed. An architecture for distributed enterprise data mining. In Peter Sloot, Marian Bubak, Alfons Hoekstra, and Bob Hertzberger, editors, *High-Performance Computing and Networking*, volume 1593 of *Lecture Notes in Computer Science*, pages 573–582. Springer Berlin - Heidelberg, 1999.

[30] 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*, 8(6):911–922, December 1996.

[31] Jacques Cohen. Bioinformatics An Introduction for Computer Scientists. *ACM Computing Surveys*, 36(2):122–158, June 2004.

[32] F. Corradini, L. Mariani, and E. Merelli. A programming environment for global activity-based applications. In Giuliano Armano, Flavio De Paoli, Andrea Omicini, and Eloisa Vargiu, editors, *Proceedings of the 4^{th} AI*IA/TABOO Joint Workshop "From Objects to Agents": Intelligent Systems and Pervasive Computing*, pages 163–169, 10-11 September 2003.

[33] F. Corradini, L. Mariani, and E. Merelli. An agent-based approach to tool integration. *International Journal on Software Tools for Technology Transfer*, 6(3):231–244, August 2004.

[34] F. Corradini and E. Merelli. Hermes: Agent-Based Middleware for Mobile Computing, Formal Methods for Mobile Computing. In Marco Bernardo and Alessandro

Bogliolo, editors, *Formal Methods for Mobile Computing*, volume 3465 of *Lecture Notes in Computer Science*, pages 234–270. Springer-Verlag, 2005.

[35] F. Corradini, E. Merelli, and M. Vita. A multi-agent system for modelling carbohydrate oxidation in cell. In Osvaldo Gervasi, Marina L. Gavrilova, Vipin Kumar, Antonio Lagan, Heow Pueh Lee, Youngsong Mun, David Taniar, and Chih Jeng Kenneth Tan, editors, *Computational Science and Its Applications - ICCSA 2005*, volume 3481 of *Lecture Notes in Computer Science*, pages 1264–1273. Springer Berlin - Heidelberg, 2005.

[36] K. Decker, S. Khan, C. Schmidt, G. Situ, R. Makkena, and D. Michaud. Biomas: A multi-agent system for genomic annotation. *International Journal of Cooperative Information Systems*, 11(3-4):265–292, 2002.

[37] K. Decker, X. Zheng, and C.J. Schmidt. A Multi-Agent System for Automated Genomic Annotation. In *Proceedings of the 5th International Conference on Autonomous Agents(AGENTS01)*, pages 433–440, Montreal, Quebec, Canada, May 28-June 1 2001. ACM.

[38] M. Delgado, W. Fajardo, E. Gibaja, and R. Perez-Perez. BioMen: An information system to herbarium. *Expert Systems with Applications*, 28(3):507–518, 2005.

[39] P. Dey, T. Gatton, and M. Amin. Modeling Human Computer Interactions with Automata. *Transactions on Computers*, 3(1):187–189, January 2004.

[40] S. Dilyana and G. Petya. Building distributed applications with Java mobile agents. International e-Business Plan Tutorial Appendix D: SOFTWARE (INTELLIGENT) AGENTS, 2002.

[41] Zoheir Ezziane. Applications of artificial intelligence in bioinformatics : A review. *Expert Systems with Applications*, 30:2–10, 2006.

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

[43] Stan Franklin and Art Graesser. Is It an agent, or just a program?: A taxonomy for autonomous agents. In Jrg P. Mller, Michael J. Wooldridge, and Nicholas R. Jennings, editors, *Intelligent Agents III Agent Theories, Architectures, and Languages*, volume 1193 of *Lecture Notes in Computer Science*, pages 21–35. Springer Berlin - Heidelberg, 1997.

[44] Yongjian Fu. Distributed Data Mining: An Overview. Department of Computer Science, University of Missouri-Rolla, 2001.

[45] Paolo Gamba, Olaf Hellwich, and Pierfrancesco Lombardo. Fusion of remotely sensed data over urban areas. *Information Fusion*, 6(3):189–192, September 2005.

[46] Lei Gao, Hua Dai, Tong-Liang Zhang, and Kuo-Chen Chou. Remote Data Retrieval for Bioinformatics Applications: An Agent Migration Approach. *PLoS ONE*, 6(6):1–7, 2011.

[47] A.J. Gibbs and G.A. McIntyre. The diagram, a method for comparing sequences. *European Journal of Biochemistry*, 16:1–11, 1970.

[48] Jorge J. Gmez-Sanz, Marie-Pierre Gervais, and Gerhard Weiss. A Survey on Agent-Oriented Oriented Software Engineering Research. In Federico Bergenti, Marie-Pierre Gleizes, and Franco Zambonelli, editors, *Methodologies and Software Engineering for Agent Systems*, volume 11 of *Multiagent Systems, Artificial Societies, and Simulated Organizations*, pages 33–62. Springer US, 2004.

[49] J. R. Graham, K. Decker, and M. Mersic. DECAF - A Flexible Multi Agent System Architecture. *Autonomous Agents Multi-Agent Systems*, 7(1-2):7–27, 2003.

[50] Robert S. Gray, David Kotz, George Cybenko, and Daniela Rus. *Mobile Agents: Motivations and State-of-the-Art Systems*. Department of Computer Science, Dartmouth College Hanover, NH, USA, 2000.

[51] Nitin Gupta, Nitin Mangal, Kamal Tiwari, and Pabitra Mitra. Mining Quantitative Association Rules in Protein Sequences. In Graham J. Williams and Simeon J. Simoff, editors, *Data Mining*, volume 3755 of *Lecture Notes in Computer Science*, pages 273–281. Springer Berlin - Heidelberg, 2006.

[52] Cornelia Gyorodi, Robert Gyorodi, and Stefan Holban. A comparative study of association rules mining algorithms . In *Proceedings of the 1st Romanian-Hungarian Joint Symposium on Applied Computational Intelligence(SACI 2004)* , pages 213–222, May 25-26 2004.

[53] J. Han and M. Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann, 2nd edition, 2006.

[54] Jiawei Han, Jian Pei, Yiwen Yin, and Runying Mao. Mining Frequent Patterns without Candidate Generation: A Frequent-Pattern Tree Approach. *Data Mining and Knowledge Discovery*, 8(1):53–87, January 2004.

[55] Carl Hewitt. Viewing control structures as patterns of passing messages. *Artificial Intelligence*, 8(3):323–364, June 1997.

[56] Jochen Hipp, Ulrich Guntzer, and Gholamreza Nakhaeizadeh. Algorithms for association rule mining a general survey and comparison. *ACM SIGKDD Explorations Newsletter*, 2(1):58–64, June 2000.

[57] D. Hollingsworth. Workflow management coalition: The workflow reference model. Technical Report WFMC-TC-1003 Draft 1.1, Workflow Management Coalition(WfMC), November 1994.

[58] Vasant Honavar, Leslie Miller, and Johnny Wong. Distributed Knowledge Networks. In *Proceedings of the IEEE Information Technology Conference* , 1998.

[59] Gongzhu Hu and Shaozhen Ding. An Agent-Based Framework for Association Rules Mining of Distributed Data. In Roger Lee and Naohiro Ishii, editors, *Software Engineering Research, Management and Applications 2009*, volume 253 of *Studies in Computational Intelligence*, pages 13–26. Springer Berlin - Heidelberg, 2009.

[60] Renata Ivancsy, Ferenc Kovacs, and Istvan Vajk. An analysis of association rule mining algorithms. In *Proceedings of the 4th International ICSC Symposium on Engineering of Intelligent Systems*, pages 774–778, Feb. 29 - March 2 2004.

[61] R. C. Joshi and Shashikala Tapaswi. Image Similarity: A Genetic Algorithm Based Approach. *International Journal of Computer, Control, Quantum and Information Engineering*, 1(3):662–666, 2007.

[62] Murat Kantarcioglu and Chris Clifton. Privacy-Preserving Distributed Mining of Association Rules on Horizontally Partitioned Data. *IEEE Transactions on Knowledge and Data Engineering*, 16(9):1026–1037, September 2004.

[63] K.A. Karasavvas, R. Baldock, and A. Burger. Bioinformatics integration and agent technology. *Journal of Biomedical Informatics*, 37(3):205–219, June 2004.

[64] H. Kargupta, B. Park, D. Hershberger, and E. Johnson. Collective Data Mining: A new perspective toward Distributed Data Mining. In Hillol Kargupta and Philip Chan, editors, *Advances in Distributed and Parallel Knowledge Discovery*, pages 131–178. AAAI/MIT Press, 1999.

[65] Hillol Kargupta and Philip Chan. *Advances in Distributed and Parallel Knowledge Discovery*. AAAI/MIT Press, August 2000.

[66] Hillol Kargupta, Ilker Hamzaoglu, and Brian Stafford. Scalable, Distributed Data Mining Using An Agent Based Architecture . In David Heckerman, Heikki Mannila, Daryl Pregibon, and Ramasamy Uthurusamy, editors, *Proceedings of the 3rd International Conference on the Knowledge Discovery and Data Mining(KDD-97)*, pages 211–214. AAAI Press, Menlo Park, California, 1997.

[67] J.W. Keele and J. Wray. Software agents in molecular computational bioloy. *Briefing in Bioinformatics*, 6(4):370–379, 2005.

[68] M. Klusch, S. Lodi, and M. Gianluca. The role of agents in distributed data mining: issues and benefits . In *Proceedings of the IEEE/WIC International Conference on Intelligent Agent Technology(IAT 2003)*, pages 211–217. IEEE, 13-16 Oct. 2003.

[69] Matthias Klusch. Information agent technology for the Internet: a survey. *Data and Knowledge Engineering*, 36(3):337–372, March 2001.

[70] Shonali Krishnaswami. *A Hybrid Model for Delivering Internet-based Distributed Data Mining Services* . PhD thesis, School of Computer Science and Software Engineering, Monash University, Australia, November 2002.

[71] U. P. Kulkarni, P. D. Desai, Tanveer Ahmed, J. V. Vadavi, and A. R. Yardi. Mobile Agent Based Distributed Data Mining. In *Proceedings of the International Conference on Computational Intelligence and Multimedia Applications(ICCIMA 2007)*, pages 18–24. IEEE Computer Society, 2007.

[72] H.C. Lam, M.V. Garcia, B. Juneja, S. Fahrenkrug, and D. Boley. A Multi-agent Approach to Gene Expression Analysis. In *Proceedings of 2nd International Workshop*

*on Multi-Agent Systems for Medicine, Computational Biology, and Bioinformatics (MAS*BIOMED'06)*, pages 60–73, Hakodate, Japan, May 9 2006. Future University.

[73] Danny B. Lange and Mitsuru Oshima. *Programming and deploying Java mobile agents with Aglets*. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA, First edition, 1998.

[74] Arthur M. Lesk. *Introduction to Bioinformatics*. Oxford University Press, Fourth edition, February 2014.

[75] Loredana Lo Conte, Steven E. Brenner, Tim J. P. Hubbard, Cyrus Chothia, and Alexey G. Murzin. SCOP database in 2002: refinements accommodate structural genomics. *Nucleic Acids Research*, 30(1):264–267, Jan 2002.

[76] M. Luck, P. McBurney, O. Shehory, and S. Willmott. Agent Technology: Computing as Interaction (A Roadmap for Agent Based Computing). AgentLink III, September 2005.

[77] M. Luck and E. Merelli. Agents in bioinformatics. *Knowledge Engineering Review*, 20(2):117–125, 2005.

[78] Alessandro Maccagnan, Tullio Vardanega, Erika Feltriny, Giorgio Valley, Mauro Rivaz, and Nicola Cannata. Multi-agent system for the automated handling of experimental protocols in biological laboratories. In A. Omicini and M. Viroli, editors, *Proceedings of the 11th Workshop on Objects and Agents(WOA 2010),Dagli Oggetti Agli Agenti*, Rimini, Italy, September 5-7 2010.

[79] Edna Marquez, Jesus Savage, Jaime Berumen, Christian Lemaitre, Ana Lilia Laureano-Cruces, Ana Espinosa, Ron Leder, and Alfredo Weitzenfeld. A Decision Support System Based on Multi-Agent Technology for Gene Expression Analysis. *International Journal of Intelligence Science*, 5:158–172, April 2015.

[80] Gale L. Martin, Amy Unruh, and Susan D. Urban. InfoSleuth: An agent infrastructure for knowledge discovery and event detection. Technical Report MCC-INSL-003-99, Microelectronics and Computer Technology Corporation (MCC)., 1999.

[81] E. Merelli, G. Armano, N. Cannata, F. Corradini, M. dInverno, A. Doms, P. Lord, A. Martin, L. Milanesi, S. Mller, M. Schroeder, and M. Luck. Agents in bioinformatics, computational and systems biology. *Briefings in Bioinformatics*, 8(1):45–59, May 2006.

[82] E. Merelli, R. Culmone, and L. Mariani. Bioagent: A mobile agent system for bioscientists. In *Network Tools and Applications in Biology(NETTAB) Workshop - Agents in Bioinformatics*, 2002.

[83] E. Merelli and M. Young. Validating MAS Models with Mutation. In *Proceedings of the 1st AAMAS International Workshop on Multi-Agent Systems for Medicine, Computational Biology, and Bioinformatics (MAS*BIOMED'05)*, pages 146–156, Utrecht,Netherlands, July 25 2005.

[84] Alexey G. Murzin, Steven E. Brenner, Tim J. P. Hubbard, and Cyrus Chothia. SCOP: a structural classification of proteins database for the investigation of sequences and structures. *Journal of Molecular Biology*, 247(4):536–540, Apr 1995.

[85] H. S. Nwana. Software Agents: An Overview. *Knowledge Engineering Review*, 11(3):1–40, September 1996.

[86] A. O. Ogunde, O. Folorunso, A. S. Sodiya, and G. O. Ogunleye. A Review of Some Issues and Challenges in Current Agent-Based Distributed Association Rule Mining. *Asian Journal of Information Technology*, 10(2):84–95, 2011.

[87] A. O. Ogunde, O. Folorunso, A. S. Sodiya, J. A. Oguntuase, and G. O. Ogunleye. Improved cost models for agent-based association rule mining in distributed databases. *Anale SEria Informatica*, 9(1):231–250, 2011.

[88] Robert Orfali, Dan Harkey, and Jeri Edwards. *The essential distributed objects survival guide*. John Wiley & Sons, USA, 1995.

[89] A. Orro, M. Saba, E. Vargiu, and G. Mancosu. Using a personalized, adaptive and cooperative multiagent system to predict protein secondary structure. In *Proceedings of the 1st AAMAS International Workshop on Multi-Agent Systems for Medicine, Computational Biology, and Bioinformatics (MAS*BIOMED'05)*, pages 170–183, Utrecht,Netherlands, July 25 2005.

[90] C. Ouzounis and A. Valencia. Early bioinformatics: the birth of a discipline - A personal view. *Bioinformatics*, 19(17):2176–2190, 2003.

[91] Byung-Hoon Park and Hillol Kargupta. Distributed Data Mining: Algorithms, Systems, and Applications. Department of Computer Science and Electrical Engineering, University of Maryland Baltimore County, 1000 Hilltop Circle Baltimore, MD 21250, 2002.

[92] Srinivasan Parthasarathy and Sandhya Dwarkadas. Shared State for Distributed Interactive Data Mining Applications. *Journal of Distributed and Parallel Databases*, 11(2):129–155, March 2002.

[93] Srinivasan Parthasarathy and Ramesh Subramonian. Facilitating Data Mining on a Network of Workstations. In Hillol Kargupta and Philip Chan, editors, *Advances in Distributed Data Mining*, pages 229–254. AAAI Press, 1999.

[94] Ram Bahadur Patel and Kumkum Garg. A flexible security framework for mobile agents. *Control and Intelligent Systems*, 33(3):175–183, September 2005.

[95] Vu Anh Pham and A. Karmouch. Mobile Software Agents: An Overview. *IEEE Communications Magazine*, 36(7):26–37, July 1998.

[96] Gian Pietro Picco. Mobile Agents: An Introduction. *Microprocessors and Microsystems*, 25(2):65–74, 2001.

[97] Paolo Romano. Automation of in-silico data analysis processes through workflow management systems. *Briefings in Bioinformatics*, 9(1):57–68, December 2007.

[98] Manoj K. Sachan, Gurpreet Singh Lehal, and Vijender Kumar Jain. A novel method to segment online gurmukhi script. In Chandan Singh, Gurpreet Singh Lehal, Jyotsna Sengupta, Dharam Veer Sharma, and Vishal Goyal, editors, *Information Systems for Indian Languages*, volume 139 of *Communications in Computer and Information Science*, pages 1–8. Springer Berlin - Heidelberg, 2011.

[99] Arun Kumar Sangaiah and Arun Kumar Thangavelu. An adaptive neuro-fuzzy approach to evaluation of team-level service climate in GSD projects. *Neural Computing and Applications*, 25(3-4):573–583, 2014.

[100] J. C. Setubal and J. Meidanis. *Introduction to Computational Molecular Biology.* PWS Publishing Company, 1997.

[101] Taeshik Shon, Jaeik Cho, Kyusuk Han, and Hyohyun Choi. Toward Advanced Mobile Cloud Computing for the Internet of Things: Current Issues and Future Direction. *Mobile Networks and Applications (MONET)*, 19(3):404–413, June 2014.

[102] K. L. Shunmuganathan, K. V. Deeba, and K. Deepika. Agent Based Bioinformatics integration Using RETSINA. *International Arab Journal of Information Technology*, 5(3):258–264, 2008.

[103] Claudio Silvestri. *Distributed and Stream Data Mining Algorithms for Frequent Pattern Discovery.* PhD thesis, Dipartimento di Informatica, Universita Ca Foscari di Venezia, Italia, 2006.

[104] Ramakrishnan Srikant and Rakesh Agrawal. Mining quantitative association rules in large relational tables . In *Proceedings of the ACM SIGMOD international conference on Management of data(SIGMOD1996)* , pages 1–12. ACM New York,USA, June 1996.

[105] R. Steven, C. Goble, P. Kaker, and Brass. A. A classification of tasks in bioinformatics. *Bioinformatics*, 17(2):180–188, 2001.

[106] Salvatore Stolfo, Andreas L. Prodromidis, Shelley Tselepis, Wenke Lee, Dave W. Fan, and Philip K. Chan. JAM: Java Agents for Meta-Learning over Distributed Databases. In David Heckerman, Heikki Mannila, Daryl Pregibon, and Ramasamy Uthurusamy, editors, *Proceedings of the 3rd International Conference on the Knowledge Discovery and Data Mining(KDD-97)*, pages 74–81. AAAI Press, Menlo Park, California, 1997.

[107] Andreas L. Symeonidis and Pericles A. Mitkas. *Agent Intelligence Through Data Mining* , volume 14 of *Multiagent Systems, Artificial Societies, and Simulated Organizations.* Springer, First edition, 2005.

[108] Arvind Kumar Tiwari and Rajeev Srivastava. A Survey of Computational Intelligence Techniques in Protein Function Prediction. *International Journal of Proteomics*, 2014(Article ID 845479), December 2014.

[109] Grigorios Tsoumakas and Ioannis Vlahavas. Distributed Data Mining. Department of Informatics, Aristotle University of Thessaloniki, Thessaloniki, Greece, 2009.

[110] Yun-Lan Wang, Zeng-Zhi Li, and Hai-Ping Zhu. Mobile agent based distributed and incremental techniques for association rules. In *Proceedings of the International Conference on Machine Learning and Cybernetics(ICMLC 2003)*, volume 1, pages 266–271, 2003.

[111] Yunlan Wang, Zengzhi Li, Jun Xue, and Yinliang Zhao. A Novel Incremental Algorithm for Mining Frequent Itemsets . In *Proceedings of the 2000 International Symposium on Distributed Computing and Application to Business Engineering and Science(DCABES 2002)* , pages 60–64, 2002.

[112] WfMC. Workflow Management Coalition: Terminology & Glossary. Technical Report WFMC-TC-1011 Issue 3.0, Workflow Management Coalition(WfMC), Feb 1999.

[113] Michael Wooldridge. Intelligent agents: The key concepts. In Vladimr Mak, Olga tpnkov, Hana Krautwurmov, and Michael Luck, editors, *Multi-Agent Systems and Applications II*, volume 2322 of *Lecture Notes in Computer Science*, pages 3–43. Springer Berlin - Heidelberg, 2002.

[114] Michael Wooldridge. *An Introduction to MultiAgent Systems*. John Wiley & Sons, second edition, May 2009.

[115] Michael Wooldridge and Nicholas R. Jennings. Intelligent Agents: Theory and Practice. *Knowledge Engineering Review*, 10(2):115–152, June 1995.

[116] Xindong Wu and S. Zhang. Synthesizing high-frequency rules from different data sources. *IEEE Transactions on Knowledge and Data Engineering*, 15(2):353–367, March-April 2003.

[117] Pengyi Yang, Li Tao, Liang Xu, and Zili Zhang. Multiagent framework for bio-data mining. In Peng Wen, Yuefeng Li, Lech Polkowski, Yiyu Yao, Shusaku Tsumoto, and Guoyin Wang, editors, *Rough Sets and Knowledge Technology*, volume 5589 of *Lecture Notes in Computer Science*, pages 200–207. Springer Berlin - Heidelberg, 2009.

[118] Qiang Yang and Xindong Wu. 10 Challenging Problems in Data Mining Research. *International Journal of Information Technology & Decision Making*, 5(4):597–604, December 2006.

[119] M. J. Zaki. Parallel and distributed association mining: a survey. *IEEE Concurrency*, 7(4):14–25, 1999.