EFFICIENT MINING OF FREQUENT COLOSSAL ITEMSETS FROM HIGH DIMENSIONAL DATA

Thesis

Submitted in partial fulfilment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

by

Mr. Manjunath K Vanahalli

DEPARTMENT OF INFORMATION TECHNOLOGY

NATIONAL INSTITUTE OF TECHNOLOGY KARNATAKA

SURATHKAL, MANGALORE - 575025

March 2020

Declaration

I hereby *declare* that the Research Thesis entitled "Efficient Mining of Frequent Colossal Itemsets from High Dimensional Data" which is being submitted to the National Institute of Technology Karnataka, Surathkal in partial fulfilment of the requirements for the award of the Degree of Doctor of Philosophy in Information Technology is a *bonafide report of the research work carried out by me*. The material contained in this thesis has not been submitted to any University or Institution for the award of any degree.

Mr. Manjunath K Vanahalli
Register No.: 145063IT14F02
Department of Information Technology

Place: NITK Surathkal
Date:

Certificate

This is to *certify* that the Research Thesis entitled “Efficient Mining of Frequent Colossal Itemsets from High Dimensional Data” submitted by Mr. Manjunath K Vanahalli (Register Number: 145063IT14F02) as the record of the research work carried out by him, is *accepted as the Research Thesis submission* in partial fulfilment of the requirements for the award of degree of Doctor of Philosophy.

Dr. Nagamma Patil
Research Guide
Assistant Professor
Department of Information Technology
NITK Surathkal - 575025

Chairman - DRPC
(Signature with Date and Seal)

Acknowledgements

First and foremost, I would like to extend my sincere gratitude to my research guide, Dr. Nagamma Patil, for her dedicated help, advice, inspiration, encouragement, enthusiasm, and continuous support, throughout my research career.

I express my sincere thanks to members of my RPAC committee, Dr. Shashidhar G Koolagudi, Dept. of CSE and Dr. R. Madhusudhan, Dept. of MACS, for their valuable feedback and constructive suggestions during my research work.

I thank all of Department of Information Technology, NITK - all faculties and staff for numerous opportunity of learning and all the support at times of need. I thank NITK Surathkal as a whole for providing me with necessary platform for my research and for the opportunity of attaining my Ph.D.

I thank my wonderful Parents - Suma K Vanahalli and Krishna G Vanahalli, for the love, patience, care and moral support they have showered on me throughout my life, even more importantly, during the period of my research. I would like to thank my Sister - Anita Lokesh and Brother-in-law Lokesh G for their support and care.

I thank my wonderful Wife - Shylaja for the love, patience, care and moral support she has showered on me. I thank my Mother-in-law - Uma Devi and her family for their support and care. I thank all my family members for their prayers and support.

I express my heartfelt thanks to my friends - Sanjay S Bankapur, Shridhar G Domanal, Karthik N, Ashwin T S, Gokul S Krishnan, Girish G N, Ranjit P Kolkar, Archana Bhat and Shruti J R for their support and care. I thank all my close friends who has been a great support during the period of my research. I also thank my labmates, fellow research scholars, ALs and non-teaching staff for their support.

(Mr. Manjunath K Vanahalli)

Abstract

The basic and major step of Association Rule Mining (ARM) is itemset mining. ARM and itemset mining have a great and vast range of applications. The conventional featured enumeration based itemset mining algorithms focus on mining frequent itemsets, frequent closed itemsets, and frequent maximal itemsets from transactional datasets. The transactional datasets consist of a smaller number of attributes (features) and a large number of rows (samples). The abundant data across a variety of domains, including bioinformatics has led to the formation of a new form of dataset known as high dimensional dataset, whose data characteristics are different from that of transactional datasets. The high dimensional datasets consist of a large number of features and a smaller number of rows. The amount of information that can be extracted from high dimensional datasets is potentially huge, but extraction of information from these datasets is a non-trivial task. The result of Frequent Itemset Mining (FIM) and Frequent Closed Itemset Mining (FCIM) algorithms include small and mid-sized itemsets, which do not enclose valuable and complete information for decision making. In applications dealing with high dimensional datasets such as bioinformatics, ARM gives greater importance to the large-sized itemsets known as colossal itemsets.

The recent research focused on mining frequent colossal itemsets and frequent colossal closed itemsets, which are more influential in decision making and are significant for many applications, especially in the field of bioinformatics. The preprocessing technique of existing frequent colossal itemset mining and frequent colossal closed itemset mining algorithms fail to prune the complete set of insignificant features and rows. An Effective Improved Preprocessing (EIP) technique has been proposed to prune the complete set of insignificant features and rows, which confines an increase in the mining search space. The existing frequent colossal itemset mining algorithm mine limited set of frequent colossal itemsets leading to the generation of an incomplete set of association rules, which consequently affects the decision making. Frequent colossal itemset mining algorithm has been proposed to achieve better accuracy than existing algorithms in terms of mining number of frequent colossal itemsets from the high dimensional dataset.

The existing algorithms for mining Frequent Colossal Closed Itemsets (FCCI) from the high dimensional dataset do not enclose an efficient pruning strategy and closeness checking method. To overcome the drawbacks of the existing works, an algorithm enclosed with efficient Rowset Cardinality Table (RCT) based closeness checking method and pruning strategy has been proposed to efficiently mine FCCI from high dimensional dataset.

The existing algorithms are inefficient in mining FCCI from the datasets consisting of a large number of features and rows, as they are inefficient in handling the changing characteristics of data subset during the mining process. The combination of different enumeration methods is required to efficiently handle different characteristics possessed by different datasets. A dynamic switching algorithm has been proposed to efficiently mine FCCI form the dataset consisting of a large number of features and rows. The dynamic switching algorithm efficiently handles the changing characteristics of the data subset during the mining process. The dynamic switching algorithm is enclosed with Itemset Support Table (IST) based closeness checking method and pruning strategy.

The existing algorithms for mining FCCI from high dimensional datasets are sequential and computationally expensive. Distributed and parallel computing is a good strategy to overcome the inefficiency of the existing sequential algorithms. The inefficiency of the existing sequential algorithms has been overcome by proposing the parallel row enumerated algorithm to efficiently mine FCCI from the high dimensional dataset. Traversing the row enumerated tree is the best solution for mining FCCI from the high dimensional dataset. The intrinsic nature of the row enumerated tree is typically unbalanced, as the number of nodes in each row enumerated tree branch vary. The distributed and parallel algorithm with load balancing has been designed to address the inefficiency of existing works.

Keywords: Bioinformatics; High Dimensional Dataset; Data Characteristics; Preprocessing; Frequent Colossal Itemsets; Frequent Colossal Closed Itemsets; Rowset Cardinality Table; Itemset Support Table; Dynamic Switching; Pruning Strategy; Closeness Checking; Parallel algorithm; Load Balancing.

Contents

List of Tables

List of Figures

List of Abbreviations

ARM	Association Rule Mining
FCI	Frequent Closed Itemsets
FMI	Frequent Maximal Itemsets
FCCI	Frequent Colossal Closed Itemsets
RNA	Ribo Nucleic Acid
FP-growth	Frequent-Pattern growth
FP-tree	Frequent-Pattern tree
PC-tree	Pattern Count tree
LOFP-tree	Lexicographically Ordered Frequent Pattern tree
AFOP-tree	Ascending Frequency Ordered Prefix tree
CP-tree	Compact Pattern tree
PIETM	Principle of Inclusion-Exclusion and Transaction Mapping
GA-Apriori	Genetic Algorithm based Apriori
PSO-Apriori	Particle Swarm Optimization based Apriori
BTP	Balanced Tidset Parallel
CARM	Cloud-Based Association Rule Mining
HD-mine	High Distributed mine
FD-mine	Fast Distributed mine
DFIMA	Distributed Frequent Itemset Mining Algorithm
P2S	Parallel-2-Steps
PATD	Parallel Absolute Top Down
HPC	High Performance Computing
SS	Single Scan
GPU	Graphics Processing Unit
LCM	Linear time Closed itemset Miner
DBV-Miner	Dynamic Bit Vector Miner
CFSP-Miner	Closed Frequent Similar Pattern Miner
DFCIM	Distributed Frequent Closed Itemsets Mining
FR-tree	Frequent Row Tree
IP-List	Itemset Pointer List
VTD	Vertical Top-Down
FCCI	Frequent Colossal Closed Itemsets
PF	Pattern Fusion
CPM	Colossal Pattern Miner
CR-tree	Compact-Row tree

CP-tree	Colossal Pattern tree
KDD	Knowledge Discovery in Databases
FIM	Frequent Itemset Mining
FCIM	Frequent Closed Itemset Mining
FMIM	Frequent Maximal Itemset Mining
FCCIM	Frequent Colossal Closed Itemset Mining
EIP	Effective Improved Preprocessing
minsup	*minimum support threshold*
mincard	*minimum cardinality threshold*
PT	Prune Table
SFCCIM	Set of Frequent Colossal Closed Itemsets
RCT	Rowset Cardinality Table
BSFCCIM	BitSet Frequent Colossal Closed Itemset Mining
MLL	Mixed Lineage Leukemia
DLBCL	Diffuse Large B-Cell Lymphoma
DSFCCIM	Dynamic Switching Frequent Colossal Closed Itemset Mining
IST	Itemset Support Table

Chapter 1

Introduction

In the following sections, a brief introduction, motivation for the present study, and the research contributions has been explained.

1.1 Knowledge Discovery in Databases

Rapid development in information technology has provided the organization with the ability to store, process, and retrieve a huge amount of data. Nevertheless, there is a need to extract useful information from the data in an effective and efficient manner, which helps the experts from different domains to make necessary decisions. This has led to the importance of data mining and the necessity to provide efficient and effective associated algorithms.

Knowledge Discovery in Databases (KDD) is the process of discovering useful knowledge from a collection of data. The widely used data mining technique is a process that includes data preparation, selection, data cleansing, and incorporating prior

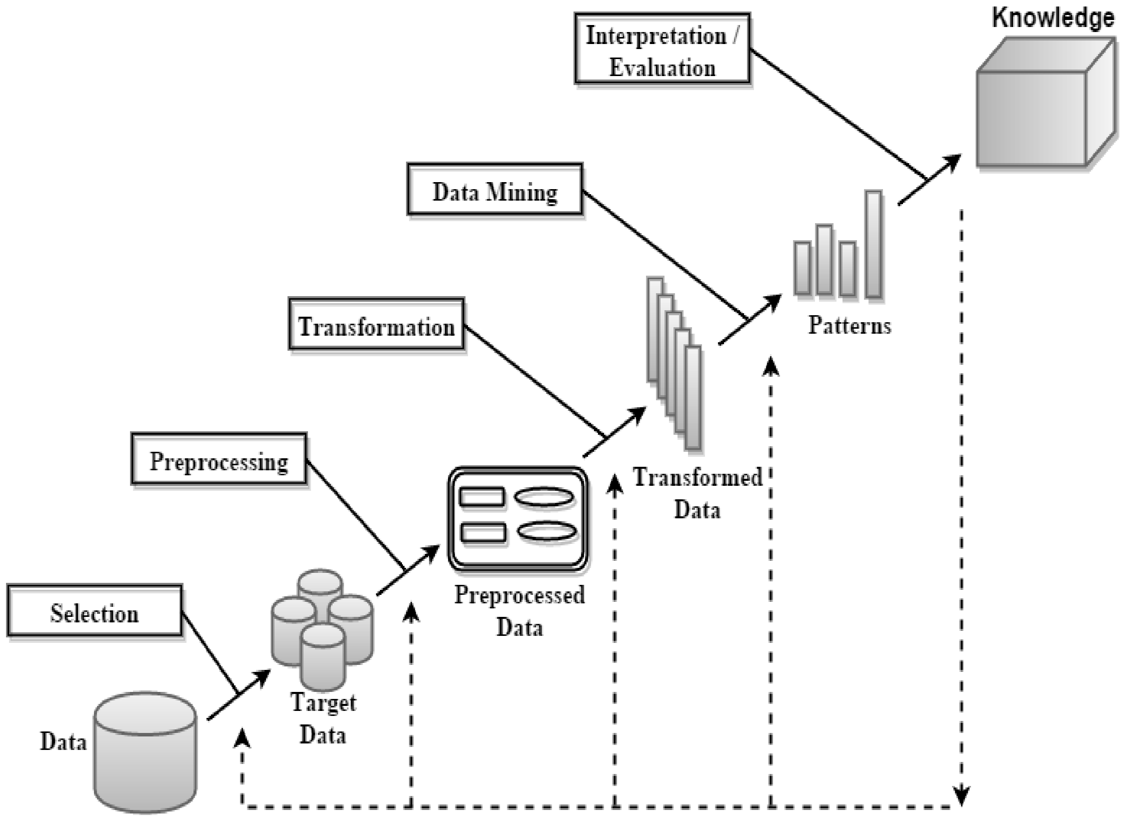


Figure 1.1. Steps in the KDD Process.

knowledge on datasets and interpreting accurate solutions from the observed results. Data mining is the analysis step of KDD and helps to extract useful information from the collection of data (Fayyad *et al.* (1996)). Extracted useful information assists the domain experts in decision making. Figure 1.1 summarizes the steps that compose the KDD process (Fayyad *et al.* (1996), Jiawei Han and Pei (2011)).

1.2 Data Mining Tasks

There are numerous data mining tasks, and these tasks can be classified as follows:

- **Summarization** is the abstraction or generalization of the data. A set of task relevant data is summarized and abstracted, resulting in a smaller collection that gives a general overview of the data.

- **Classification** is the derivation of a function or model which determines the class of an object based on its attributes. The function or model is derived based on the analysis of a set of training data.

- **Association** is the discovery of togetherness or connection of objects. Such a kind of togetherness or connection is termed as association rules. An association rule reveals the associative relationship between the objects.

- **Clustering** is the process of grouping a set of data objects into multiple groups or clusters so that objects within a cluster have high similarity, but very dissimilar to the objects in the other clusters.

1.3 Association Rule Mining and Itemset Mining

Association Rule Mining (ARM) is one of the important data mining tasks which has been well recognized over the past two decades. Itemset mining is the major and fundamental part of ARM. Figure 1.2 highlights the different types of itemset mining. Frequent Itemset Mining (FIM), Frequent Closed Itemset Mining (FCIM), Frequent Maximal Itemset Mining (FMIM), Frequent Colossal Itemset Mining, Frequent Colossal Closed Itemset Mining (FCCIM) are the different types of itemset mining. Many algorithms have been designed for mining frequent itemsets from the transactional dataset. The compressed forms of the frequent itemsets such as Frequent Closed Itemsets (FCI) and Frequent Maximal Itemsets (FMI) were proposed due to the generation of redundant rules from a large number of mined frequent itemsets. FMI are the lossy compression of frequent itemsets. Hence the research focused on mining FCI rather than mining

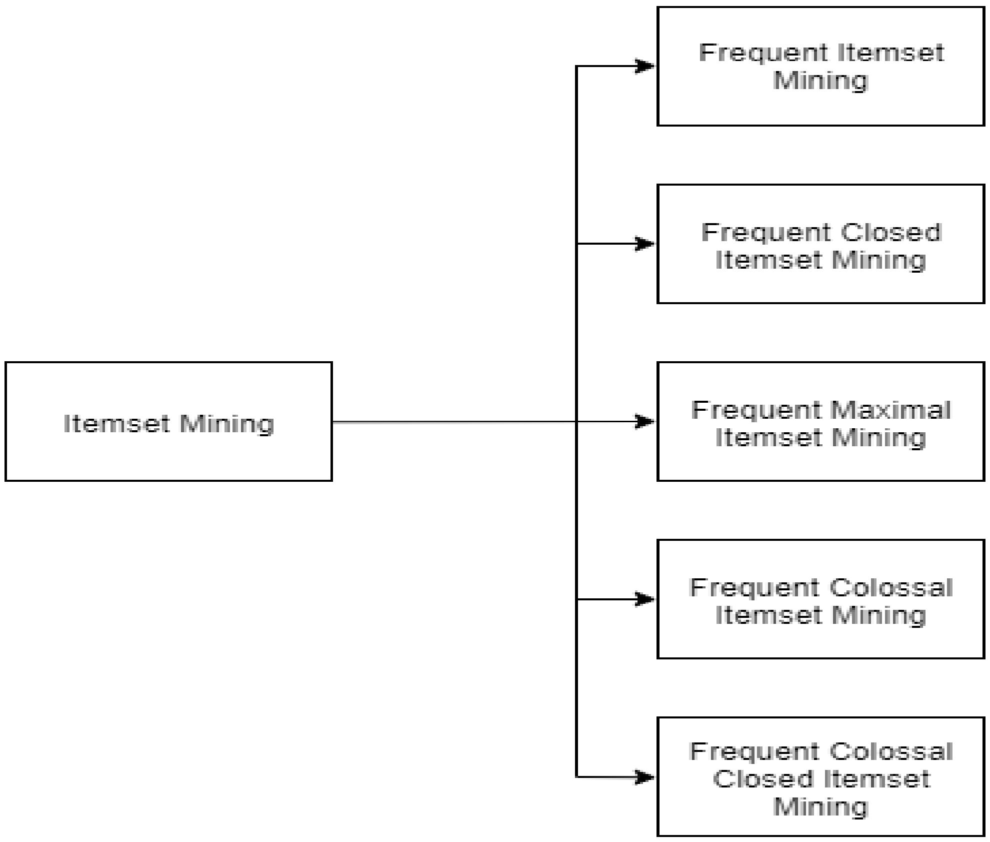

Itemset Mining
Frequent Itemset Mining
Frequent Closed Itemset Mining
Frequent Maximal Itemset Mining
Frequent Colossal Itemset Mining
Frequent Colossal Closed Itemset Mining

Figure 1.2. Different Types of Itemset Mining.

FMI. FCI are lossless compression of the frequent itemsets as there is no information loss from the set of association rules generated by FCI.

Mining FCI has received great interest over the past two decades. Conventional algorithms focus on mining FCI from transactional datasets consisting of a large number of rows (samples) and a smaller number of attributes (features). These conventional algorithms are feature enumeration based algorithms as they tend to mine FCI by searching the itemset space. An average increase in the transaction length leads to an exponential increase in the running time of these algorithms. In the modern era, abundant data across a variety of domains, including bioinformatics, has led to the formation of high dimensional datasets, whose data characteristics are different from that of transactional datasets. These high dimensional datasets have a smaller number of rows and a considerably large number of features. The amount of information that can be extracted from high dimensional datasets is potentially huge, but the extraction of information from these datasets is a non-trivial task.

The conventional algorithms face an uphill task in mining FCI from the high dimensional dataset. The computational problems of these conventional algorithms were solved by proposing row enumerated algorithms to mine FCI from the high dimensional dataset. These row enumerated algorithms tend to mine FCI by searching the rowset space. The result of FCI mining algorithms includes a large number of small and mid-sized itemsets, which does not enclose valuable and complete information in many applications. In the applications dealing with high dimensional datasets such as bioinformatics, ARM gives greater importance to the large-sized itemsets called as colossal itemsets. The colossal itemsets are more influential in decision making and are significant in many applications. Alves *et al.* (2009) and Naulaerts *et al.* (2015) showed the importance of discovering the colossal itemsets from high dimensional datasets such as gene expression data. In bioinformatics, the strong associations between mined FCCI helps in building the associative classifier for achieving higher classification accuracy. Further, the associations between the mined FCCI will be used in gene expression data analysis to uncover the gene networks. The existing FIM algorithms and FCIM algorithms are inefficient in mining frequent colossal itemsets and Frequent Colossal Closed Itemsets (FCCI) from the high dimensional dataset as they expend an exponential time in mining a large number of small and mid-sized itemsets.

Zhu *et al.* (2007) were the first to introduce the concept of colossal itemsets and designed the Pattern Fusion (PF) algorithm to mine them. The existing FCIM algorithms prune the insignificant features before proceeding with the mining of FCI. The insignificant features and insignificant rows have to be pruned before proceeding with the mining of frequent colossal itemsets and FCCI. The existing frequent colossal itemset mining algorithms and FCCI mining algorithms do not enclose an effective preprocessing technique. The preprocessing technique used in the existing algorithms fails to prune the complete set of insignificant features and insignificant rows, which leads to an increase in the mining search space. Algorithms like Pattern Fusion (Zhu *et al.* (2007)) and BVBUC (Sohrabi and Barforoush (2012)) mine limited set of frequent colossal itemsets and FCCI leading to the generation of an incomplete set of association rules, which consequently affects the decision making. Most of the mined frequent colossal itemsets and FCCI by the BVBUC algorithm tend to provide incorrect support information leading to the generation of an incorrect set of association rules, resulting in deficient decision making. The existing FCCI mining algorithms are inefficient in handling the changing characteristics of the data subset during the mining process. Also, the pruning strategies to cut down the row and feature enumerated search space are inefficient. The closeness checking method of the rowset and itemset enclosed by the existing algorithms are inefficient. This highlights the inefficiency of these existing algorithms in mining FCCI from the high dimensional dataset.

1.4 Research Motivation

A typical business transaction dataset for market basket analysis has a relatively large number of rows (transactions) compared to a smaller number of columns (features). However, other application areas such as gene expression matrices analysis in bioinformatics, text mining, combinatorial chemistry, and multivariate imaging involve a different form of dataset known as a high dimensional dataset. The data characteristics of the high dimensional datasets are different from that of transactional datasets. The high dimensional datasets have a smaller number of rows and a considerably large number of features, while the transactional datasets have a large number of rows and a considerably smaller number of features. The high dimensional datasets have attracted interest from researchers to devise a new method to extract significant and important information efficiently. The amount of information that can be extracted from high dimensional datasets is potentially huge, but extraction of information and knowledge from these datasets is a non-trivial task.

Applications that deal with high dimensional datasets include: Discovering relationships between the data values within gene expression matrices or microarray datasets, in order to assist in understanding the cause and effect of biological processes. Such relationships can help in generating gene networks. In the field of bioinformatics, association rules play a significant role in reporting biological relevant associations between environmental conditions and genes, and between different genes. They also provide biological information about genes and gene expressions. ARM gives greater importance to the large-sized itemsets called as colossal itemsets. The colossal itemsets are more influential in decision making and are significant in many applications. Hence it is important to mine colossal itemsets from the high dimensional dataset.

1.5 Preliminaries

Let the high dimensional dataset *D* (R,F) consist of *m* number of rows, R = $\{r_1, r_2, r_3,...., r_m\}$ and *n* number of features, F = $\{f_1, f_2, f_3,..., f_n\}$. Each r_i consists of a set of features and has a unique row identifier *rid*. A non-empty subset of features, $X \subseteq$ F is defined as an itemset. An itemset consisting of *l* features is defined as *l*-itemset. Let r(f_j) signify the rows in which the j^{th} feature of the dataset is present. A non-empty subset of *rid*s, Y $\subseteq$ R is defined as a rowset. A rowset consisting of *l rid*s is defined as *l*-rowset. Let f(r_i) signify the features occurring in the i^{th} row of the dataset.

Example 1. *Table 1.1 shows an example of a high dimensional dataset D consisting of*

8 rows, where each row is described with unique row identifier (rid), R= {1, 2, 3, 4, 5, 6, 7, 8} and 11 features, F= {a, b, c, d, e, f, g, h, i, j, k}.

Definition 1 (Support)**.** *The number of rows in which an itemset X occurs is called the support of an itemset, denoted by sup(X).*

Example 2. *In Table 1.1, the support of an itemset X= {b, d, g, h}, sup(X) is 2.*

Definition 2 (Support Set)**.** *The rows in which an itemset X occurs is called support set of an itemset, denoted by supset(X).*

Example 3. *In Table 1.1, the support set of an itemset X= {b, d, g, h}, supset(X) is 23.*

Definition 3 (Cardinality)**.** *The number of items in an itemset X is known as the cardinality of an itemset, denoted by card(X).*

Example 4. *In Table 1.1, the cardinality of an itemset X= {b, d, g, h}, card(X) is 4.*

Definition 4 (Frequent Itemset)**.** *An itemset X is called frequent itemset if and only if sup(X) $\geq$ minsup, where minsup is user specified least support threshold.*

Example 5. *In Table 1.1, the itemset X= {b, h} is frequent itemset with minimum support threshold set to 2, sup(X) $\geq$ 2.*

Definition 5 (Frequent Closed Itemset)**.** *An itemset X is called frequent closed itemset if and only if it is frequent and there exists no proper superset X$''$, (X $\subset$ X$''$) such that support of X is same as the support of X$''$, sup(X)=sup(X$''$).*

Table 1.1. High Dimensional Dataset *D*

row id (*rid*)	features
1	*a, b, d, f, j*
2	*a, b, d, g, h*
3	*b, d, g, h*
4	*a, b, f, h, i, j*
5	*a, c, d, g, h, j*
6	*b, d, i*
7	*e, g*
8	*e, k*

Example 6. *In Table 1.1, the itemset X= {d, g, h} is frequent closed itemset with minimum support threshold set to 2 because dgh is frequent and there exists no proper superset X″ with the same support as X.*

Definition 6 (Frequent Colossal Itemset)**.** *An itemset X is called frequent colossal itemset if and only if it is frequent and card(X) ≥ mincard, where mincard is user specified least cardinality threshold.*

Example 7. *In Table 1.1, the itemset X= {a, b, f, j} is frequent colossal itemset with minimum support threshold set to 2 and minimum cardinality threshold set to 4, sup(X) ≥ 2 and card(X) ≥ 4.*

Definition 7 (Frequent Colossal Closed Itemset)**.** *An itemset X is called frequent colossal closed itemset if and only if it is frequent closed and card(X) ≥ mincard, where mincard is user specified least cardinality threshold.*

Example 8. *In Table 1.1, the itemset X= {b, d, g, h}, is frequent colossal closed itemset with minimum support threshold set to 2 and minimum cardinality threshold set to 4, sup(X) ≥ 2 and card(X) ≥ 4.*

Definition 8 (Closure)**.** *Given an itemset X ⊆ F and a rowset Y ⊆ R in a high dimensional dataset D (R,F), we define*

$$r(X) = \{r_i \in R \mid \forall f_j \in X,\ f_j\ is\ present\ in\ r_i\ of\ D\} \tag{1.1}$$

$$f(Y) = \{f_j \in F \mid \forall r_i \in Y,\ f_j\ is\ present\ in\ r_i\ of\ D\} \tag{1.2}$$

The closure of an itemset X, C(X) and closure of a rowset Y, C(Y) is defined as follows

$$C(X) = f(r(X)) \tag{1.3}$$

$$C(Y) = r(f(Y)) \tag{1.4}$$

1.6 Major Contributions of the Thesis

The salient contributions of the research work are listed as follows:

- An Effective Improved Preprocessing (EIP) technique has been proposed to prune the complete set of insignificant features and insignificant rows from the high dimensional dataset by effective utilization of minimum support threshold (*minsup*) and minimum cardinality threshold (*mincard*) respectively.

- Frequent colossal itemset mining algorithm has been proposed to achieve better accuracy than existing algorithms in terms of mining number of frequent colossal itemsets from the high dimensional dataset.
- An efficient Rowset Cardinality Table (RCT) based closeness checking method has been proposed to check the closeness of a rowset during the row enumeration method. An efficient RCT based pruning strategy has been proposed to cut down the row enumerated mining search space.
- The algorithm integrated with an efficient RCT based closeness checking method and pruning strategy has been proposed to mine the complete set of FCCI from the high dimensional dataset.
- An efficient Itemset Support Table (IST) based closeness checking method has been proposed to check the closeness of an itemset during the feature enumeration method. An efficient IST based pruning strategy has been proposed to cut down the feature enumerated mining search space.
- The dynamic switching algorithm integrated with efficient closeness checking methods and pruning strategies has been proposed to efficiently mine FCCI from the dataset consisting of a large number of features and a large number of rows. The dynamic switching algorithm efficiently handles the changing characteristics of the data subset during the mining process.
- The parallel row enumerated algorithm has been proposed to efficiently mine FCCI from the high dimensional dataset.
- The distributed and parallel algorithm with load balancing has been designed to address the unbalanced intrinsic nature of the row enumerated tree.

1.7 Organization of the Thesis

Chapter 2 describes a survey of related literature on the techniques used for mining different types of itemset mining, followed by the problem statement and research objectives.

Chapter 3 briefly highlights the proposed effective improved preprocessing technique.

The mining of frequent colossal itemsets from the high dimensional dataset and the performance of the proposed frequent colossal closed itemset mining algorithm enclosed with RCT based closeness checking method and pruning strategy is explained in chapter 4.

Chapter 5 discusses the dynamic switching algorithm for mining FCCI from the dataset consisting of a large number of rows and a large number of features.

The parallel row enumerated algorithm for mining FCCI from the high dimensional dataset is highlighted in chapter 6. Chapter 6 also describes about handling the unbalanced intrinsic nature of the row enumerated tree.

Conclusions and a look into the future directions are presented in chapter 7.

1.8 Summary

This chapter describes about knowledge discovery in databases and the steps involved in the KDD process. The data mining tasks, such as summarization, classification, association, and clustering, have been briefly discussed in this chapter. Also, the different types of itemset mining and the importance of association rule mining have been explained. The preliminaries related to the field of itemset mining have been discussed with examples. The research contributions are listed out in this chapter. The chapter also highlights about the organization of the thesis.

In the next chapter, the literature survey related to the field of itemset mining has been presented.

Chapter 2

Literature Survey

In this chapter some of the major existing works in the area of itemset mining has been reviewed. The chapter begins with section 2.1, which highlights the great and vast range of applications of itemset mining and Association Rule Mining (ARM). They are used to address the unique problems across a wide variety of data domains.

Section 2.2 focuses on sequential mining of frequent itemsets from the transactional dataset. Many efficient sequential Frequent Itemset Mining (FIM) algorithms were developed over the years. The inefficiency of sequential FIM algorithms in handling the large transactional datasets and the exponential increase in the running time due to the average increase in the transactional length led to the development of distributed and parallel FIM algorithms.

Section 2.3 highlights the distributed and parallel mining of frequent itemsets from the transactional dataset. FIM algorithms on a cluster, grid computing systems, cloud computing environment and Hadoop clusters were developed to overcome the inefficiency of sequential FIM algorithms. The researchers also developed parallel and distributed FIM algorithms using MapReduce, Spark and by exploiting the parallelism in the Graphics Processing Unit (GPU). The generation of redundant association rules from a large number of mined frequent itemsets resulted in the proposal of Frequent Closed Itemsets (FCI) and Frequent Maximal Itemsets (FMI).

Section 2.4 emphasizes on sequential mining of FCI from the transactional dataset. The problem of generating redundant association rules was handled by developing efficient sequential algorithms to mine FCI. The research focused on mining FCI rather than FMI, which is the lossy compression of frequent itemsets. The sequential algorithms are inefficient in mining FCI from the large transactional dataset; this led to the designing of parallel and distributed frequent closed itemset mining algorithms. Section 2.5 highlights the parallel and distributed mining of FCI from the large transactional dataset.

The conventional parallel and sequential algorithms developed for mining FCI from the transactional dataset face an uphill task in mining FCI from the high dimensional dataset due to its data characteristics. The inefficiency and the uphill task of these algorithms were overcome by designing row enumerated algorithms to mine FCI from the high dimensional dataset; this has been emphasized in section 2.6. The result of FCI mining algorithms includes very large number of small and mid-sized itemsets, which do not enclose the valuable and complete information in many applications. In applications dealing with high dimensional datasets such as bioinformatics, ARM gives greater importance to the large-sized itemsets known as colossal itemsets. Section 2.7 exhibits the mining of frequent colossal itemsets and Frequent Colossal Closed Itemsets (FCCI) from the high dimensional dataset.

2.1 Applications of Itemset Mining and Association Rule Mining

FIM has a great and vast range of applications, including customer analysis, software bug detection, web analysis, event detection, spatiotemporal analysis, text analysis, toxicological analysis, Ribo Nucleic Acid (RNA) analysis and chemical compound predication (Aggarwal (2014); Amancio (2015*a*,*b*); Chen *et al.* (2017); Li *et al.* (2001); Naulaerts *et al.* (2015); Parsons *et al.* (2004); Silva *et al.* (2016); Viana *et al.* (2013); Xue *et al.* (2016); Yin and Han (2003); Zhong *et al.* (2012)). FIM helps in mining crucial motifs in a variety of biological and chemical applications (Aggarwal (2014); Naulaerts *et al.* (2015)). It is useful in designing methods for clustering high dimensional data (Aggarwal (2014); Parsons *et al.* (2004)). The associative classification, which is the integration of ARM and classification helps in the prediction of the subcellular location of proteins and in achieving high accuracy (Naulaerts *et al.* (2015); Yoon and Lee (2012)). ARM plays a vital role in providing visual representations of the underlying text collection (Aggarwal (2014); Li *et al.* (2001); Yin and Han (2003)). The ARM has been extensively used in gene expression data analysis to uncover the gene networks (Naulaerts *et al.* (2015)).

2.2 Sequential Mining of Frequent Itemsets from the Transactional Dataset

Many sequential algorithms have been designed in the last two decades for mining frequent itemsets from the transactional dataset. Agrawal *et al.* (1994) proposed three algorithms namely, Apriori, AprioriTID, and AprioriHybrid for mining frequent itemsets. *Generate and Test* approach has been utilized by AprioriTID and Apriori algorithms, where l-itemsets are used to *generate and test* (l+1)-itemsets. These two algorithms utilize Apriori property, which states that all nonempty subsets of a frequent itemset must also be frequent. The interesting feature of AprioriTID algorithm compared to the Apriori algorithm is that the transactional dataset is not utilized for calculating the support after the first pass. The AprioriHybrid algorithm is designed by selecting the best features of AprioriTID and Apriori algorithm. The AprioriHybrid algorithm switches from Apriori to AprioriTID algorithm after a specific number of passes depending on the count of the candidate set.

The disadvantage of a generation of a large number of candidate sets in *generate and test* approach and repeated scans of the transactional dataset by Apriori algorithm led to the pattern growth approach. Han *et al.* (2000) were the first to design an algorithm to mine frequent itemsets based on *pattern growth* approach. The Frequent-Pattern growth (FP-growth) algorithm efficiently mines the frequent itemsets by three techniques: (i) by compressing the transactional dataset in smaller and highly condensed data structure, (ii) by averting the costly formation of huge number of candidate sets by adopting the Frequent-Pattern tree (FP-tree) based mining, and (iii) the mining task are decomposed into group of smaller task to mine the itemsets from conditional dataset by following the partition based divide and conquer method.

Ananthanarayana *et al.* (2000) proposed the Pattern Count tree (PC-tree). The PC-tree is constructed with a single transactional dataset scan, and it can be dynamically updated. PC-tree portrays the compact and complete transactional dataset. PC-tree helps in the generation of Lexicographically Ordered Frequent Pattern tree (LOFP-tree), which is a unique form of an ordered tree. Liu *et al.* (2003*a*) proposed a frequent itemset mining algorithm by using the condensed Ascending Frequency Ordered Prefix tree (AFOP-tree). The AFOP-tree is used to organize the conditional datasets, and it also helps in saving the space by storing the single branches in arrays. The algorithm uses the top-down search strategy to traverse the AFOP-tree to mine the frequent itemsets.

Liu *et al.* (2003*b*) revisited the problem of mining frequent itemsets from the transactional dataset. The authors discussed four different dimensions which help to increase the efficiency of the mining; (i) the itemset search order can be either ascending frequency order or lexicographical order, (ii) the conditional dataset can be represented by either array based structure or tree based structure, (iii) the construction strategy of conditional dataset can be either pseudo or physical, (iv) the tree is traversed either by top-down or bottom-up search strategy. Qiu *et al.* (2004) proposed an efficient QFP-growth algorithm, which inherits the advantages of FP-growth and avoids the bottleneck of creating a large number of conditional FP-trees. The QFP-growth avoids the creation of a large number of conditional FP-tress by constructing the dynamic temporary root to improve its efficiency.

Dong and Han (2007) designed BitTableFI algorithm for mining frequent itemsets. This algorithm uses the BitTable data structure vertically and horizontally to condense the transactional dataset for fast support count and generation of candidate itemsets. Index-BitTableFI algorithm was designed by Song *et al.* (2008). Corresponding computing method and index array were proposed by the authors to utilize the BitTable horizontally. The implementation of hybrid search helped in reducing the search space to a great extent.

Tanbeer *et al.* (2009) presented a Compact Pattern tree (CP-tree), a novel tree data structure whose mining performance is the same as that of FP-growth. The CP-tree captures the information about the transaction dataset by one scan. The dynamic tree restructuring concept was introduced by CP-tree to generate a highly compact frequency-descending tree structure. Lin *et al.* (2014) proposed the Principle of Inclusion-Exclusion and Transaction Mapping (PIETM) algorithm. The PIETM algorithm uses the feature enumerated bottom-up search strategy for mining frequent itemsets. PIETM algorithm uses the Inclusion-Exclusion principle to determine the support of the candidate itemsets instead of scanning the dataset. Mapping and storing the transaction *id*(s) are done in an interval list which provides appropriate information for mining frequent itemsets.

Aryabarzan *et al.* (2018) proposed the NegNodeset data structure whose basis is the same as the set of nodes in a prefix tree. The proposed negFIN algorithm utilizes a NegNodeset data structure for mining frequent itemsets. Bitmap representation based encoding model is employed by the NegNodeset data structure for prefix tree nodes. The negFIN algorithm mines the frequent itemsets by employing a feature set enumeration tree and prunes the tree search space by using a promotion method.

Djenouri and Comuzzi (2017) proposed a framework for mining frequent itemset using bio-inspired approaches which examine the recursive property of frequent itemsets. The combination of the bio-inspired stochastic search process and recursive property of frequent itemsets is considered for efficiently traversing the itemset search space. The framework involved bio-inspired algorithms such as Genetic Algorithm based Apriori (GA-Apriori) algorithm and Particle Swarm Optimization based Apriori (PSO-Apriori) algorithm for mining frequent itemsets. The section describes about sequential FIM algorithms. These sequential FIM algorithms face an uphill task if the user-specified minimum support threshold is set very low. The inefficiency of sequential FIM algorithms in handling the large transactional datasets and the exponential increase in the running time due to the average increase in the transactional length led to the development of parallel and distributed FIM algorithms.

2.3 Parallel and Distributed Mining of Frequent Itemsets from Transactional Datasets

The section illustrates the parallel and distributed mining of frequent itemsets from the transactional dataset. Javed and Khokhar (2004) proposed the FP-growth based parallel FIM algorithm for shared nothing multiprocessor platforms or message passing systems. The proposed algorithm does not explicitly replicate the entire counting data structure on each processor. The algorithm efficiently partitions the FP-tree and the list of frequent elements among the processors; this helps in synchronization and also helps in reducing the communication overheads.

Zhou and Yu (2008*b*) and Yu and Zhou (2010) designed transaction set based parallel algorithm for mining frequent itemsets on clusters and grid computing system. FP-tree inspired transaction identification based parallel FP-tree is utilized by the parallel algorithm to mine the frequent itemsets from the transactional dataset. The execution time is efficiently decreased by reducing both the tree insertion cost and communication cost. Zhou and Yu (2008*a*) proposed FP-tree based parallel and distributed balanced transactions set algorithm to mine frequent itemsets from the transactional dataset on the grid computing system. The transactions are efficiently exchanged by using the transaction identification set instead of scanning the dataset. The Balanced Tidset Parallel (BTP) algorithm considers the tree width and depth to balance the load on the grid computing system.

Lin and Deng (2010) designed a parallel and distributed Cloud-Based Association Rule Mining (CARM) algorithm. The CARM algorithm preserves the data privacy and efficiently utilizes the nodes in a cloud computing environment to mine frequent itemsets. High Distributed mine (HD-mine) and Fast Distributed mine (FD-mine) are the two parts of the proposed CARM algorithm. HD-Mine handles the complex mining task by coordinating the available nodes in the computing environment, and FD-mine is used for quick mining of frequent itemsets from transactional datasets. Lin and Lo (2013) proposed four algorithms to handle the many task computing issues rather than boosting the performance of the single task. The authors proposed CARM inspired algorithms to mine frequent itemsets and to provide reliable, scalable and fast mining service in many task computing environments.

Zhang *et al.* (2015) proposed a distributed algorithm using spark for mining frequent itemsets. The amount of candidate itemsets is significantly reduced by Distributed Frequent Itemset Mining Algorithm (DFIMA) with the help of matrix-based approach. To enhance the iterative computational efficiency, the DFIMA has been implemented using a memory-based distributed framework such as a spark. The matrix-based pruning technique adopted by the DFIMA significantly helps in reducing the number of dataset scans.

Salah *et al.* (2017) discussed the parallel FIM problem for very large datasets. The authors also highlighted the effectiveness and impact of adopting data placement strategies in a distributed environment. Parallel-2-Steps (P2S) and Parallel Absolute Top Down (PATD) are the highly scalable parallel FIM algorithms proposed by the authors. Simple and efficient parallel jobs of the P2S algorithm help in mining frequent itemsets from the large transactional dataset. The mining of frequent itemsets from the large dataset is kept very compact and simple by PATD algorithm. The PATD algorithm is equipped with one parallel job, which helps in reducing the communication cost, running time and the overhead of energy power consumption in a massively distributed environment.

Djenouri *et al.* (2018) address the FIM problem by employing the High Performance Computing (HPC) approach. The authors proposed a Single Scan (SS) algorithm and three HPC versions of the algorithm. The first HPC version of SS algorithm called as GSS was implemented on Graphics Processing Unit (GPU) architecture by efficiently mapping the input data and thread blocks. The second HPC version of SS algorithm called as CSS was implemented on a cluster computing environment by scheduling the workers to independent jobs. The third HPC version of SS algorithm called as CGSS was implemented using multiple GPU cluster nodes, which helps in accelerating the mining process. The authors proposed three partitioning strategies to scale down the imbalance among the cluster nodes and the GPU thread divergence.

Xun *et al.* (2016) and Xun *et al.* (2017) designed two parallel algorithms for mining frequent itemsets from the transactional dataset. The Map-Reduce programming model has been utilized to design the FiDoop parallel algorithm for mining the frequent itemsets. Ultra-metric tree of the frequent itemsets has been incorporated by the FiDoop algorithm to avoid the generation of conditional pattern base and to have compressed storage. The mining tasks are completed with the help of three MapReduce jobs. The FiDopp-HD parallel algorithm was designed by the authors to improve the mining speedup for large datasets. The FiDoop-HD is an extension of the FiDoop algorithm. The mining and communication overhead problem was addressed by the FiDoop-HD algorithm.

Chon *et al.* (2018) designed an algorithm to mine the frequent itemsets from the large scale dataset. The GMiner algorithm is a GPU-based fast parallel algorithm. The GPU's computational power is explored by the parallel GMiner algorithm to achieve fast mining performance. The itemsets are mined from the initial level of the feature enumeration tree, this counter intuitive way of performing the mining tasks improves the efficiency of the algorithm. An array consisting of relative memory addresses is split to handle the workload skewness problem effectively.

The generations of redundant association rules from a large number of mined frequent itemsets resulted in the proposal of Frequent Closed Itemsets (FCI) and Frequent Maximal Itemsets (FMI). The research focused on mining FCI rather than FMI, which is the lossy compression of frequent itemsets. Frequent closed itemsets are lossless compression of the frequent itemsets as there is no information loss from the set of association rules generated by frequent closed itemsets.

2.4 Sequential Mining of Frequent Closed Itemsets from the Transactional Dataset

This section emphasizes on sequential mining of FCI from the transactional dataset. The problem of generating redundant association rules was handled by developing efficient sequential algorithms to mine FCI. Pasquier *et al.* (1999) were the first to propose Frequent Closed Itemsets (FCI). The closure mechanism was utilized to design a Frequent Closed Itemset Mining (FCIM) algorithm called as A-Close. Galois connection-based closure mechanism was used by an A-Close algorithm. Frequent itemset mining problem was reduced to frequent closed itemsets mining problem as a reduced set of association rules are generated without information loss.

Pei *et al.* (2000) designed a CLOSET algorithm for mining FCI from the transactional dataset. Three techniques were enclosed with CLOSET algorithm (i) FCIM without *generate and test* approach, but with the help of compressed structure called a Frequent Pattern tree (FP-tree). (ii) Quick identification of FCI with the help of single prefix path compression technique. (iii) Scalability of the algorithm to mine FCI from the large database is achieved by exploring a partition-based projection mechanism.

Wang *et al.* (2003) proposed CLOSET+ algorithm for mining FCI from the transactional dataset. The CLOSET+ algorithm is the extension of the CLOSET algorithm. The extensive study of different strategies by the authors led to the design of CLOSET+ algorithm. The authors highlight the pros and cons of different strategies such as physical vs. pseudo-projection of the conditional database, horizontal vs. vertical formats, tree vs. other data structure, depth-first vs. breadth-first search, top-down vs. bottom-up traversal.

Zaki and Hsiao (2002) and Zaki and Hsiao (2005) designed an efficient CHARM algorithm for mining FCI from the transactional dataset. An efficient hybrid dual itemset tidset search tree helps the CHARM algorithm to skip the feature enumeration levels. During the computation, the non-closed sets are removed by the algorithm with the help of a fast hash-based approach. The authors also designed the CHARM-L algorithm to provide a frequent closed lattice. The frequent closed lattice is helpful in the generation of rules and their visualization.

Uno *et al.* (2003), Uno *et al.* (2004) and Uno *et al.* (2005) designed the three versions of the Linear time Closed itemset Miner (LCM) algorithm. The mined FCI are used to establish the tree-shaped traversal routes. The parent-child relationship is defined between the FCI in the LCM algorithm (Uno *et al.* (2003)). The LCM second version (Uno *et al.* (2004)) was designed to improve the performance of LCM first version algorithm. The LCM second version algorithm is additionally enclosed with the pruning strategy and database reduction technique. Different data structures have disadvantages and advantages depending on the transactional dataset from which the FCI has to be mined. The combination data structures like array list, prefix tree, and the bitmap has been utilized to design the third version of the LCM algorithm (Uno *et al.* (2005)).

Lucchese *et al.* (2006) designed the scalable algorithm called as a DCI_CLOSED algorithm for mining FCI from the transactional dataset. The divide and conquer approach is adopted by the algorithm. The algorithm also exploits the bitwise vertical representation of the transactional dataset. Lexicographic order is not followed by the DCI_CLOSED algorithm for mining the FCI. The DCI_CLOSED algorithm is enclosed with an efficient pruning strategy. The Dynamic Bit Vector Miner (DBV-Miner) was proposed by Vo *et al.* (2012). Dynamic bit vector approach is used by the DBV-Miner algorithm for improving the computation. The pruning strategy enclosed by the algorithm utilizes the subsumption concept. The lookup table provides the details for the faster computation of an itemset support.

Fumarola *et al.* (2016) proposed the CloFAST algorithm for mining FCI. Vertical identifier list and sparse identifier list play a vital role in deciding the data representation of the dataset. The support count of the FCI is efficiently counted by considering the theoretical properties of the vertical identifier list and sparse identifier list. They also help in the closure checking of itemsets and pruning the mining search space. Rodríguez-González *et al.* (2018) proposed the Closed Frequent Similar Pattern Miner (CFSP-Miner) algorithm. The tree consisting of a complete set of the closed frequent similar patterns is utilized by the CFSP-Miner algorithm. The tree is defined by the parent-child relationship. CFSP-Miner algorithm is enclosed with an efficient pruning strategy to snip down the mining search space.

The sequential FCIM algorithms face an uphill task, if the user specified minimum support threshold is set very low. The inefficiency of sequential FCIM algorithms in handling the large transactional datasets and the exponential increase in the running time due to the average increase in the transactional length led to the development of parallel and distributed FCIM algorithms.

2.5 Parallel and Distributed Mining of Frequent Closed Itemsets from Transactional Datasets

The section illustrates the parallel and distributed mining of FCI from the transactional dataset. Lucchese *et al.* (2007) designed the first parallel algorithm for mining FCI from the transactional dataset. The parallel MT_CLOSED algorithm works with the multi-threading concept. The FCIM problem is decomposed into many independent tasks to achieve parallelization. The load misbalancing problem is addressed by considering both the dynamic and static scheduling policy.

Fu and Foghlu (2008) designed the first distributed algorithm for mining FCI from the large transactional dataset. Frequent closed mining search space is partitioned into several independent non-overlapping subspaces. These subspaces are utilized to mine the FCI. The algorithm depends upon the density priority to mine FCI. Liu *et al.* (2007) accomplished the FCIM problem in the distributed environment. The authors designed the Distributed Frequent Closed Itemsets Mining (DFCIM) algorithm for mining FCI with the exact support count. The controlling of the mined FCI is done through the Frequent Closed Itemset trees (FCItrees) data structure.

Negrevergne *et al.* (2010) proposed a parallel LCM algorithm for mining FCI from the transactional dataset. To efficiently achieve dynamic work sharing, the parallel algorithm utilizes the tuple space, which is a powerful parallelism interface. Tuple space concept-based Melinda, which is a parallel environment was presented by the authors. Melinda helps in the utilization of computation distribution models which are internally very efficient. Wang *et al.* (2012) designed the Map-Reduce based parallel AFOPT-close algorithm for mining the FCI from the large transaction dataset. An efficient parallel closure method was presented by the authors to check the closeness of the global frequent itemsets. Filtering the redundant itemsets was done on the basis of a new definition of local closed itemsets and global closed itemsets.

The incremental approach based parallel and distributed algorithm was proposed by Sreedevi *et al.* (2014) for mining FCI from the large transactional dataset. The vertical data format has been utilized by the algorithm. The communication cost between the processors has been efficiently reduced by the algorithm. Parallel and generic PARAMINER algorithm was designed by Negrevergne *et al.* (2014) for mining FCI. The feature enumeration mining search space is explored by the PARAMINER algorithm. For efficient parallel execution of the PARAMINER algorithm on the multi-core architecture, the authors designed a novel technique for the dataset reduction.

The conventional sequential algorithms discussed in section 2.4 and conventional parallel algorithms discussed in this section focus on mining FCI from transactional datasets consisting of a large number of rows (samples) and a smaller number of features (attribute). These conventional parallel and sequential algorithms are feature enumeration based algorithms as they tend to mine FCI by searching the itemset space. There will be an exponential increase in the running time of these conventional parallel and sequential algorithms as the average transaction length increases. In the modern era, the abundant data across a variety of domains, including bioinformatics have led to the new form of dataset known as a high dimensional dataset, whose data characteristics are different from that of transactional datasets. The high dimensional datasets have a smaller number of rows and a considerably large number of features. The amount of information that can be extracted from high dimensional datasets is potentially huge, but extraction of information and knowledge from these datasets is a non-trivial task. The conventional parallel and sequential algorithms face an uphill task in mining FCI from the high dimensional dataset. The inefficiency and the uphill task of these algorithms were overcome by developing row enumerated algorithms to mine FCI from the high dimensional dataset.

2.6 Mining of Frequent Closed Itemsets from the High Dimensional Dataset

This section emphasizes on the mining of FCI from the high dimensional dataset. Efficient row enumerated algorithms were designed by the researches to handle different data characteristics of the high dimensional datasets compared to the transactional datasets. Pan *et al.* (2003) designed the first row enumerated algorithm for mining FCI from the high dimensional dataset. Row enumerated CARPENTER algorithm was designed to handle the data characteristics of the high dimensional datasets. The CARPENTER algorithm is enclosed with a pruning strategy to snip the row enumerated mining search space. The transposed table of the high dimensional dataset is utilized by the row enumerated algorithm. The lexicographic order plays a vital role in the systematic search for the closed itemsets. The CARPENTER algorithm utilizes the bottom-up row enumerated tree for mining FCI. The X conditional transposed table is generated at every row enumerated node.

Pan *et al.* (2004) designed a combination of feature and row enumerated COBBLER algorithm for mining FCI from the dataset consisting of a large number of features and a large number of rows. Data characteristics during the mining process are considered by the COBBLER algorithm to switch between row enumeration method and feature enumeration method. The mining process is made efficient by processing each portion of the dataset using the most suitable enumeration method. The transposed table concept is utilized by the dynamic switching algorithm. The X conditional transposed table is generated at every feature and row enumerated node.

Cong *et al.* (2004) proposed RERII and REPT algorithms for mining FCI from the microarray dataset. These two efficient algorithms mine the FCI by exploring the row enumerated mining search space. Vertical representation of the data is considered by the RERII for mining FCI, while REPT consider the concept of FP-tree. RERII and REPT algorithms are enclosed by an efficient pruning strategy. Liu *et al.* (2006) designed the first top-down row enumerated algorithm for mining FCI from the high dimensional dataset. The TD-Close algorithm efficiently prunes the row enumerated mining search space by utilizing a user specified minimum support threshold. The TD-Close algorithm is enclosed with an efficient transposed table based closure checking method to check the closeness of itemsets.

Liu *et al.* (2009) designed a TTD-Close algorithm to enhance the performance of the TD-Close algorithm. An efficient data structure is designed by the authors to improve the performance of the algorithm. The trace-based closure method has been enclosed in the algorithm to check the closeness of the itemsets. The divide and conquer technique has been utilized to partition the mining search space into separate subspace. The efficient pruning strategy has been enclosed by the TTD-Close algorithm to prune the row enumerated mining search space. The TTD-Close algorithm refers to the Frequent Row Tree (FR-tree) with Itemset Pointer List (IP-List) for mining FCI.

Huang *et al.* (2013) designed an efficient TBtop algorithm for mining top-k FCI from the microarray dataset. The k indicates the specified number of FCI to be mined from the microarray dataset. The TBtop algorithm utilizes the top down row enumerated breadth first search strategy for mining FCI. TBtop algorithm compresses the row enumerated tree using FR-tree and IP-list. Singh *et al.* (2014) proposed a modified version of the CARPENTER algorithm for mining FCI from the high dimensional dataset. The different data structure is considered by the authors in the modified version of the CARPENTER algorithm. This data structure helps in giving better time complexity than the CARPENTER algorithm.

Vimieiro and Moscato (2014) focused on the computational challenges of mining the disjunctive closed itemsets from the high dimensional dataset. The Disclosed algorithm mines the disjunctive closed itemsets by utilizing the top down row enumerated depth first search strategy. Disclosed algorithm shows that itemsets can be mined indirectly from sets of samples. This allows to take advantage of the scarcity of samples in microarray data sets, or any data with similar characteristics. Sohrabi and Ghods (2015) designed an efficient Vertical Top-Down (VTD) algorithm for mining FCI from the high dimensional dataset. The top down row enumerated approach of the VTD algorithm utilizes the minimum support threshold to cut down mining search space.

The result of FCI mining algorithms includes small and mid-sized itemsets, which do not enclose the valuable and complete information for decision making. In applications dealing with high dimensional datasets such as bioinformatics, ARM gives greater importance to the large-sized itemsets known as colossal itemsets. The colossal itemsets are more influential in decision making and provide more suitable information for many applications. The importance of discovering the colossal itemsets from high dimensional datasets such as gene expression data was shown by Alves *et al.* (2009) and Naulaerts *et al.* (2015).

2.7 Mining of Frequent Colossal Itemsets and Frequent Colossal Closed Itemsets from the High Dimensional Dataset

Section 2.7 exhibits the mining of frequent colossal itemsets and Frequent Colossal Closed Itemsets (FCCI) from the high dimensional dataset. The existing frequent colossal itemset mining algorithms and FCCI mining algorithms are classified into feature enumeration and row enumeration based algorithms. Feature enumeration based frequent colossal itemset mining algorithms and FCCI mining algorithms are best suited for transactional datasets due to their data characteristics. Similarly, row enumeration based frequent colossal itemset mining algorithms and FCCI mining algorithms are best for high dimensional datasets due to their data characteristics.

Zhu *et al.* (2007) proposed the concept of large cardinality itemsets called as colossal itemsets. The authors proposed the Pattern Fusion (PF) algorithm for mining frequent colossal itemsets and FCCI itemsets. Pattern Fusion traverses the tree according to the feature enumeration method. Pattern fusion algorithm randomly discovers colossal itemsets by merging the selected small cardinality frequent itemsets called as core patterns. Pattern fusion algorithm mine the large cardinality itemsets by approximating the number of colossal closed itemsets generated rather than traversing each node of the tree. Approximating the number of colossal closed itemsets generated might lead to missing some of the significant frequent colossal itemsets and FCCI. Pattern fusion algorithm will not be able to mine the complete set of frequent colossal itemsets and FCCI leading to the generation of an incomplete set of association rules, which consequently affects the decision making.

Dabbiru and Shashi (2010) proposed a Colossal Pattern Miner (CPM) algorithm for quick mining of colossal itemsets by skipping the level-wise traversal of pattern, which is exhaustive. CPM algorithm is a feature enumeration based algorithm. The new agglomerative strategy skips the mid-sized itemsets to efficiently mine the colossal itemsets. The neighbour core itemsets are merged to mine the colossal itemsets. Sohrabi and Barforoush (2012) proposed the first row enumerated BVBUC algorithm for mining frequent colossal itemsets and FCCI from the high dimensional dataset. The authors state that the largest frequent itemset of each branch in the row enumerated tree is generated at the minimum support threshold level. The BVBUC algorithm mine the itemsets from the nodes belonging to the minimum support threshold level of a row enumerated tree and prune their descendants.

BVBUC algorithm will not be able to mine complete set of frequent colossal itemsets and FCCI leading to the generation of an incomplete set of association rules, which consequently affects the decision making. Most of the mined frequent colossal itemsets and FCCI tend to provide incorrect support information leading to the generation of an incorrect set of association rules, resulting in deficient decision making. Okubo and Haraguchi (2012) designed an algorithm for mining top-*N* frequent colossal itemsets. A clique in pattern graph with a certain condition is referred to the mined colossal itemsets. The depth-first branch and bound method is the main base for the algorithm. This is the first algorithm designed to mine the top-*N* colossal itemsets. Zhu (2014) discussed the importance of mining FCCI from the high dimensional dataset.

Zulkurnain *et al.* (2012) proposed a pure row enumeration based DisClose algorithm to mine FCCI from the high dimensional dataset. The DisClose algorithm uses row enumerated Compact-Row tree (CR-tree) data structure to mine FCCI. The algorithm refers to the itemset generator, rowset generator, and unique rowset generator for checking the closeness of an itemset. The transposed table concept is utilized by the DisClose algorithm. Prasanna and Seetha (2015) developed a DPMine algorithm for mining colossal itemset sequences from biological datasets. Doubleton itemsets are effectively mined by the DPMine algorithm. The DPT+ is enriched with doubleton itemsets. D-struct, an integrated data structure has been utilized by the DPMine algorithm. The top-down feature enumeration tree has been referred as DPT+ tree.

Nguyen *et al.* (2016) and Nguyen *et al.* (2017*b*) solved the colossal itemset mining problem by designing the Colossal Pattern tree (CP-tree), CP-Miner and PCP-Miner algorithm. Colossal itemsets are efficiently mined by referring an efficient sorting strategy. The count of significant candidates and subset checking time are reduced with the help of sorting strategy. The PCP-Miner utilizes the sorting strategy to efficiently mine frequent colossal itemsets. The CP-Miner and PCP-Miner lack the capability to mine FCCI from the high dimensional dataset. The theorem has been developed by Nguyen *et al.* (2017*a*) for pruning candidate itemsets efficiently with bottom-up manner. Based on the theorem developed by the authors, an efficient algorithm has been proposed for colossal itemset mining with itemset constraints.

The preprocessing technique used in the existing frequent colossal itemset mining algorithms and FCCI mining algorithms fails to prune the complete set of insignificant features and insignificant rows, leading to an increase in the feature and row enumerated mining search space. The existing FCCI mining algorithms are sequential and adopt the sequential row enumeration based approach. Moreover, the closeness checking methods enclosed by the existing FCCI mining algorithms to check the closeness of an itemset and rowset are in efficient. The pruning strategies enclosed by the existing FCCI mining algorithms are inefficient in cutting down the feature and row enumerated mining search space. This highlights the inefficiency of these algorithms in mining FCCI from the high dimensional dataset. Table 2.1 highlights the comparison of itemset mining algorithms.

Table 2.1. Comparison of Itemset Mining Algorithms.

Begin of Comparison							
Paper	Feature Enumeration	Row Enumeration	Suitable for Transaction Dataset	Suitable for High Dimensional Datasets	Sequential Approach	Parallel / Distributed Approach	Suitable for Mining
Agrawal *et al.* (1994)	Yes	No	Yes	No	Yes	No	Frequent Itemsets
Han *et al.* (2000)	Yes	No	Yes	No	Yes	No	✓
Ananthanarayana *et al.* (2000)	Yes	No	Yes	No	Yes	No	✓
Liu *et al.* (2003*a*)	Yes	No	Yes	No	Yes	No	✓
Liu *et al.* (2003*b*)	Yes	No	Yes	No	Yes	No	✓
Qiu *et al.* (2004)	Yes	No	Yes	No	Yes	No	✓
Dong and Han (2007)	Yes	No	Yes	No	Yes	No	✓
Song *et al.* (2008)	Yes	No	Yes	No	Yes	No	✓
Tanbeer *et al.* (2009)	Yes	No	Yes	No	Yes	No	✓
Lin *et al.* (2014)	Yes	No	Yes	No	Yes	No	✓
Aryabarzan *et al.* (2018)	Yes	No	Yes	No	Yes	No	✓

Continuation of Table 2.1

Paper	Feature Enumeration	Row Enumeration	Suitable for Transaction Dataset	Suitable for High Dimensional Datasets	Sequential Approach	Parallel / Distributed Approach	Suitable for Mining
Djenouri and Comuzzi (2017)	Yes	No	Yes	No	Yes	No	✓
Javed and Khokhar (2004)	Yes	No	Yes	No	No	Yes	✓
Zhou and Yu (2008*b*)	Yes	No	Yes	No	No	Yes	✓
Zhou and Yu (2008*a*)	Yes	No	Yes	No	No	Yes	✓
Lin and Deng (2010)	Yes	No	Yes	No	No	Yes	✓
Yu and Zhou (2010)	Yes	No	Yes	No	No	Yes	✓
Lin and Lo (2013)	Yes	No	Yes	No	No	Yes	✓
Zhang *et al.* (2015)	Yes	No	Yes	No	No	Yes	✓
Xun *et al.* (2016)	Yes	No	Yes	No	No	Yes	✓
Salah *et al.* (2017)	Yes	No	Yes	No	No	Yes	✓
Djenouri *et al.* (2018)	Yes	No	Yes	No	No	Yes	✓
Chon *et al.* (2018)	Yes	No	Yes	No	No	Yes	✓
Xun *et al.* (2017)	Yes	No	Yes	No	No	Yes	✓
Pasquier *et al.* (1999)	Yes	No	Yes	No	Yes	No	FCI

Continuation of Table 2.1

Paper	Feature Enumeration	Row Enumeration	Suitable for Transaction Dataset	Suitable for High Dimensional Datasets	Sequential Approach	Parallel / Distributed Approach	Suitable for Mining
Pei *et al.* (2000)	Yes	No	Yes	No	Yes	No	✓
Zaki and Hsiao (2002)	Yes	No	Yes	No	Yes	No	✓
Wang *et al.* (2003)	Yes	No	Yes	No	Yes	No	✓
Zaki and Hsiao (2005)	Yes	No	Yes	No	Yes	No	✓
Uno *et al.* (2003)	Yes	No	Yes	No	Yes	No	✓
Uno *et al.* (2004)	Yes	No	Yes	No	Yes	No	✓
Uno *et al.* (2005)	Yes	No	Yes	No	Yes	No	✓
Lucchese *et al.* (2006)	Yes	No	Yes	No	Yes	No	✓
Vo *et al.* (2012)	Yes	No	Yes	No	Yes	No	✓
Fumarola *et al.* (2016)	Yes	No	Yes	No	Yes	No	✓
Rodríguez-González *et al.* (2018)	Yes	No	Yes	No	Yes	No	✓
Lucchese *et al.* (2007)	Yes	No	Yes	No	No	Yes	✓
Fu and Foghlu (2008)	Yes	No	Yes	No	No	Yes	✓
Liu *et al.* (2007)	Yes	No	Yes	No	No	Yes	✓

Continuation of Table 2.1

Paper	Feature Enumeration	Row Enumeration	Suitable for Transaction Dataset	Suitable for High Dimensional Datasets	Sequential Approach	Parallel / Distributed Approach	Suitable for Mining
Negrevergne *et al.* (2010)	Yes	No	Yes	No	No	Yes	✓
Wang *et al.* (2012)	Yes	No	Yes	No	No	Yes	✓
Sreedevi *et al.* (2014)	Yes	No	Yes	No	No	Yes	✓
Negrevergne *et al.* (2014)	Yes	No	Yes	No	No	Yes	✓
Pan *et al.* (2003)	No	Yes	No	Yes	Yes	No	✓
Pan *et al.* (2004)	Yes	Yes	No	Yes	Yes	No	✓
Cong *et al.* (2004)	No	Yes	No	Yes	Yes	No	✓
Liu *et al.* (2006)	No	Yes	No	Yes	Yes	No	✓
Miao *et al.* (2006)	No	Yes	No	Yes	Yes	No	✓
Liu *et al.* (2009)	No	Yes	No	Yes	Yes	No	✓
Huang *et al.* (2013)	No	Yes	No	Yes	Yes	No	✓
Singh *et al.* (2014)	No	Yes	No	Yes	Yes	No	✓
Vimieiro and Moscato (2014)	No	Yes	No	Yes	Yes	No	✓

Continuation of Table 2.1

Paper	Feature Enumeration	Row Enumeration	Suitable for Transaction Dataset	Suitable for High Dimensional Datasets	Sequential Approach	Parallel / Distributed Approach	Suitable for Mining
Sohrabi and Ghods (2015)	No	Yes	No	Yes	Yes	No	✓
Zhu *et al.* (2007)	Yes	No	Yes	No	Yes	No	Frequent Colossal Itemsets and FCCI
Dabbiru and Shashi (2010)	Yes	No	Yes	No	Yes	No	Frequent Colossal Itemsets
Sohrabi and Barforoush (2012)	No	Yes	No	Yes	Yes	No	Frequent Colossal Itemsets
Zulkurnain *et al.* (2012)	No	Yes	No	Yes	Yes	No	FCCI

Continuation of Table 2.1

Paper	Feature Enumeration	Row Enumeration	Suitable for Transaction Dataset	Suitable for High Dimensional Datasets	Sequential Approach	Parallel / Distributed Approach	Suitable for Mining
Prasanna and Seetha (2015)	Yes	No	Yes	No	Yes	No	Frequent Colossal Itemsets
Nguyen *et al.* (2016)	No	Yes	No	Yes	Yes	No	Frequent Colossal Itemsets
Nguyen *et al.* (2017*b*)	No	Yes	No	Yes	Yes	No	Frequent Colossal Itemsets
Nguyen *et al.* (2017*a*)	No	Yes	No	Yes	Yes	No	Frequent Colossal Itemsets

End of Table

2.8 Research Gaps

Based on the literature review given above, the following research gaps have been identified.

- The preprocessing technique used in the existing frequent colossal itemset mining algorithms and FCCI mining algorithms fails to prune the complete set of insignificant features and insignificant rows, leading to an increase in the feature and row enumerated mining search space. Hence, there is a need for developing the preprocessing technique, which prunes the complete set of insignificant features and insignificant rows from the high dimensional dataset.
- The Pattern Fusion algorithm attempts to mine the frequent colossal itemsets by approximation method. The BVBUC algorithm attempts to mine the frequent colossal itemsets from the nodes belonging to the minimum support threshold level of row enumerated tree and prune their descendants. These algorithms fail to mine complete set of frequent colossal itemsets, as they tend to miss some of the significant frequent colossal itemsets. Most of the frequent colossal itemsets mined by BVBUC algorithm provide incorrect support information. The drawbacks can be addressed by utilizing both minimum support threshold (*minsup*) and minimum cardinality threshold (*mincard*).
- The closure methods enclosed by the existing FCCI mining algorithms such as Pattern Fusion, BVBUC and DisClose algorithm are inefficient in checking the closeness of an itemset and rowset. The pruning strategies enclosed by these existing FCCI mining algorithms are inefficient in cutting down the feature and row enumerated mining search space. Algorithms such as CARPENTER, TD-Close and TTD-Close encounter challenges in mining FCCI due to the presence of an explosive number of small and mid-sized itemsets. Hence, there is a need to develop an efficient frequent colossal closed itemset mining algorithm enclosed with efficient closeness checking methods and efficient pruning strategies.
- The existing algorithms are computationally expensive in mining FCCI from datasets that have a large number of rows and a large number of features, as these algorithms adopt either pure row enumeration approach or feature enumeration approach. Different data subsets are handled by the algorithms during the mining process. The characteristics of the data subset will change from one subset to another during the mining process. Hence, there is a need for developing a

new algorithm with combination of different enumeration methods to efficiently handle the changing characteristics of the data subset during the mining process.

- There is a greater importance to the colossal itemsets and these are critical to many applications especially in the field of bioinformatics. The problem of FIM and FCIM have been addressed by designing distributed and parallel algorithms. The state-of-the-art algorithms for mining FCCI from the high dimensional dataset are sequential and computationally expensive. Therefore, there is a need to explore the distributed and parallel approach to solve the problem of mining FCCI from the high dimensional dataset.

2.9 Problem Statement

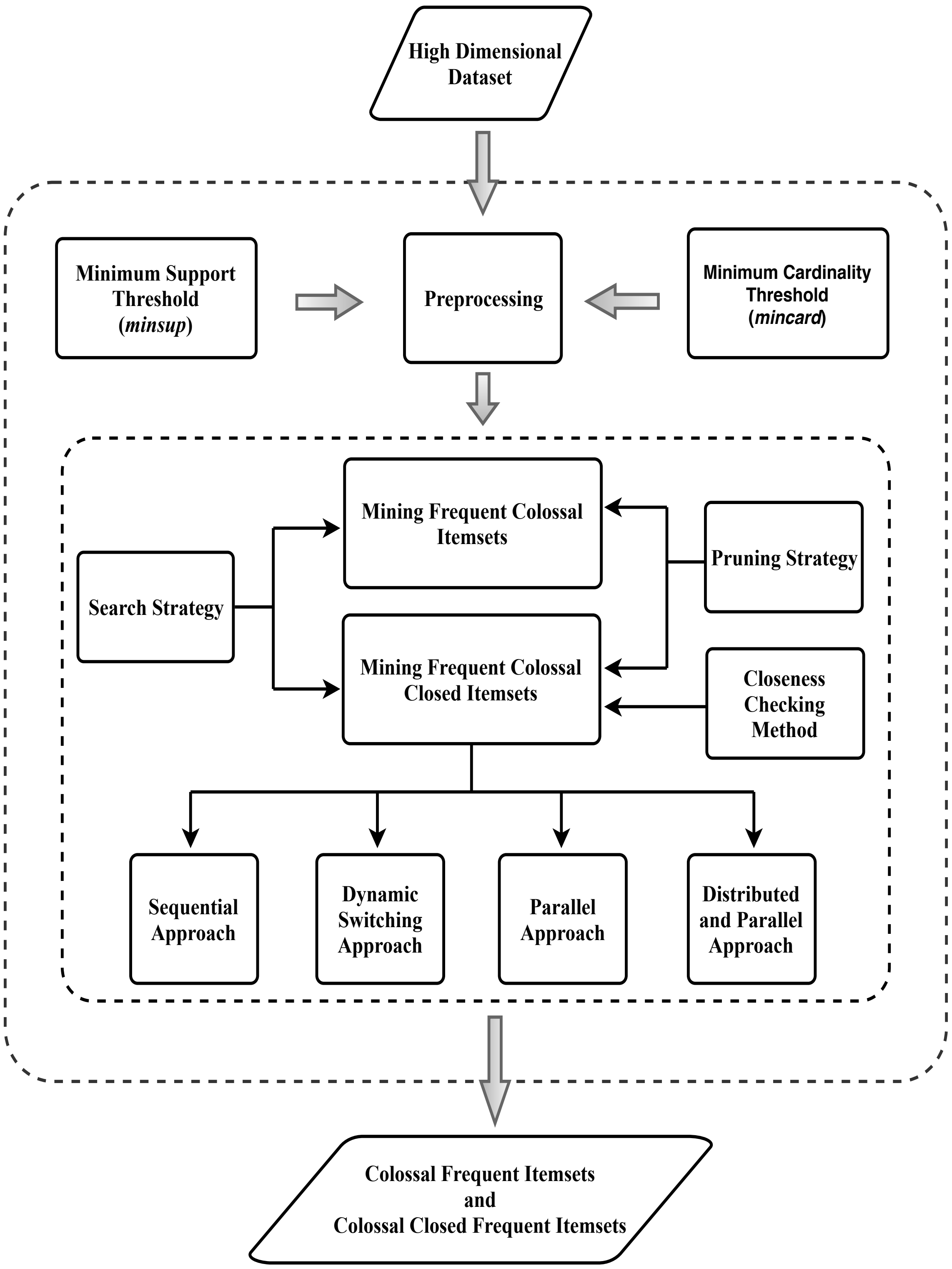
“ To design efficient algorithms for mining frequent colossal itemsets and frequent colossal closed itemsets from high dimensional datasets using efficient search and pruning strategy ”

2.10 Research Objectives

1. To design an algorithm for mining frequent colossal itemsets from high dimensional datasets.
2. To design an algorithm for mining frequent colossal closed itemsets from high dimensional datasets.
3. To design an efficient dynamic switching algorithm for mining frequent colossal closed itemsets from datasets consisting of a larger number of features and rows.
4. To design an algorithm which adopts the parallel approach to mine frequent colossal closed itemsets.
5. To improve the efficiency of parallel algorithm with efficient load balancing for mining frequent colossal closed itemsets.

2.11 Proposed Methodology

The research focuses on designing algorithms for mining frequent colossal itemsets and FCCI from the high dimensional dataset. Figure 2.1 shows the proposed methodology for mining frequent colossal itemsets and FCCI from the high dimensional dataset. The high dimensional dataset is provided as an input. An effective preprocessing technique



High Dimensional Dataset
Minimum Support Threshold (*minsup*)
Preprocessing
Minimum Cardinality Threshold (*mincard*)
Mining Frequent Colossal Itemsets
Search Strategy
Pruning Strategy
Mining Frequent Colossal Closed Itemsets
Closeness Checking Method
Sequential Approach
Dynamic Switching Approach
Parallel Approach
Distributed and Parallel Approach
Colossal Frequent Itemsets and Colossal Closed Frequent Itemsets

Figure 2.1. Proposed Methodology

has been proposed to prune the complete set of insignificant features and rows from the high dimensional dataset by effective utilization of minimum support threshold and minimum cardinality threshold respectively. It is very important to choose efficient search strategies to mine frequent colossal itemsets and FCCI. Bottom-up row enumeration approach is chosen for mining frequent colossal itemset and FCCI from the high dimensional dataset. An efficient pruning strategy has been proposed to cut down the row enumerated mining search space by efficient utilization of minimum cardinality threshold. The proposed pruning strategy provides the prior information regarding the cardinality of the itemsets to be mined at descendant row enumerated nodes without traversing them. An efficient closeness checking method has been proposed to check the closeness of rowset during the row enumeration method.

The algorithms based on either pure row or feature enumeration methods are inefficient in mining FCCI from datasets consisting of a large number of rows and a large number of features. The algorithm with a combination of different enumeration methods has been required to handle the changing characteristics of a data subset efficiently. A dynamic switching algorithm has been proposed to mine FCCI from the dataset consisting of a large number of features and rows. The dynamic switching algorithm is integrated with efficient pruning strategies, closeness checking methods to check the closeness of rowset and an itemset. It is also enclosed with efficient switching conditions that dynamically switches between bottom-up row enumeration method and bottom-up feature enumeration method to handle the changing characteristics of the data subset during the mining process. The balanced distributed parallel row enumerated algorithm has been proposed to mine FCCI from the high dimensional dataset. The load of traversing the branches of row enumerated tree among the compute nodes has been balanced by the proposed distributed and parallel algorithm.

2.12 Summary

This chapter provided a review of existing frequent itemset mining, frequent closed itemset mining, frequent colossal itemset mining and frequent colossal closed itemset mining algorithms. The problem statement and research objectives were framed based on the outcome of the literature review. The proposed methodology and a short description of the research work were presented.

In the next chapter, the proposed effective improved preprocessing technique has been discussed.

Chapter 3

Effective Improved Preprocessing Technique to Prune Insignificant features and rows from the High Dimensional Dataset

The preprocessing of the dataset is an important step in the field of itemset mining. The dataset should be preprocessed before the mining of frequent colossal itemsets and Frequent Colossal Closed Itemsets (FCCI). The preprocessing technique used in the existing frequent colossal itemset mining algorithms and FCCI mining algorithms fails to prune the complete set of insignificant features and insignificant rows. In this chapter, the proposed Effective Improved Preprocessing (EIP) technique based on minimum support threshold and minimum cardinality threshold has been discussed, which prunes the complete set of insignificant features and insignificant rows from the high dimensional dataset.

3.1 Proposed Effective Improved Preprocessing Technique

An Effective Improved Preprocessing (EIP) technique has been proposed to prune the complete set of insignificant features and insignificant rows by effective utilization of minimum support threshold (*minsup*) and minimum cardinality threshold (*mincard*), respectively. The proposed EIP technique incorporates the bitset approach for fast computation. The bitTable as shown in Table 3.1 is constructed with the same dataset characteristics as that of high dimensional dataset *D* shown in Table 1.1. If a row of high dimensional dataset *D* consists of feature f_j, then the j^{th} bit of the corresponding row in bitTable is set to 1, else it is set to 0. The *rs* in Table 3.1 indicates the number of features present in the respective row. For example, the number of features present in 5^{th} row is 6. The *cs* in Table 3.1 indicates the support of the respective feature in the bitTable. For example, the support of the feature '*d*' in bitTable is 5. The features of the high dimensional dataset which do not satisfy the criteria of *minsup* are described as insignificant features. The rows of the high dimensional dataset which do not satisfy the criteria of *mincard* are described as insignificant rows. The preprocessing technique used in the existing FCIM algorithms prunes the insignificant features before proceeding with mining of FCI. Let F′ be the set of insignificant features in the dataset. The features $\{c, k\}$ are the insignificant features with *minsup* value set to 2, F′ = $\{c, k\}$.

Table 3.1. bitTable corresponding to High Dimensional Dataset D

rid	*a*	*b*	*c*	*d*	*e*	*f*	*g*	*h*	*i*	*j*	*k*	***rs***
1	1	1	0	1	0	1	0	0	0	1	0	5
2	1	1	0	1	0	0	1	1	0	0	0	5
3	0	1	0	1	0	0	1	1	0	0	0	4
4	1	1	0	0	0	1	0	1	1	1	0	6
5	1	0	1	1	0	0	1	1	0	1	0	6
6	0	1	0	1	0	0	0	0	1	0	0	3
7	0	0	0	0	1	0	1	0	0	0	0	2
8	0	0	0	0	1	0	0	0	0	0	1	2
cs	4	5	1	5	2	2	4	4	2	3	1	

Table 3.2. bitTable after pruning insignificant features $\{c, k\}$

rid	*a*	*b*	*d*	*e*	*f*	*g*	*h*	*i*	*j*	***rs***
1	1	1	1	0	1	0	0	0	1	5
2	1	1	1	0	0	1	1	0	0	5
3	0	1	1	0	0	1	1	0	0	4
4	1	1	0	0	1	0	1	1	1	6
5	1	0	1	0	0	1	1	0	1	5
6	0	1	1	0	0	0	0	1	0	3
7	0	0	0	1	0	1	0	0	0	2
8	0	0	0	1	0	0	0	0	0	1
cs	4	5	5	2	2	4	4	2	3	

$$F_1 = (F - F') = \{a, b, d, e, f, g, h, i, j\} \tag{3.1}$$

The equation 3.1 highlights the pruning of insignificant features $\{c, k\}$ from the bitTable as shown in Table 3.1. Table 3.2 shows the bitTable after pruning the insignificant features $\{c, k\}$. The insignificant features and insignificant rows have to be pruned before proceeding with mining of frequent colossal itemsets and FCCI. Hence, the pre-processing technique in the existing colossal itemset mining algorithms Nguyen *et al.* (2017*b*, 2016, 2017*a*) and FCCI mining algorithm Zulkurnain *et al.* (2012) prune the

Table 3.3. bitTable after pruning insignificant row $\{8\}$

rid	*a*	*b*	*d*	*e*	*f*	*g*	*h*	*i*	*j*	***rs***
1	1	1	1	0	1	0	0	0	1	5
2	1	1	1	0	0	1	1	0	0	5
3	0	1	1	0	0	1	1	0	0	4
4	1	1	0	0	1	0	1	1	1	6
5	1	0	1	0	0	1	1	0	1	5
6	0	1	1	0	0	0	0	1	0	3
7	0	0	0	1	0	1	0	0	0	2
cs	4	5	5	1	2	4	4	2	3	

Table 3.4. bitTable after pruning insignificant feature $\{e\}$

rid	*a*	*b*	*d*	*f*	*g*	*h*	*i*	*j*	***rs***
1	1	1	1	1	0	0	0	1	5
2	1	1	1	0	1	1	0	0	5
3	0	1	1	0	1	1	0	0	4
4	1	1	0	1	0	1	1	1	6
5	1	0	1	0	1	1	0	1	5
6	0	1	1	0	0	0	1	0	3
7	0	0	0	0	1	0	0	0	1
cs	4	5	5	2	4	4	2	3	

insignificant rows after pruning the insignificant features. Let R′ be the set of insignificant rows from the dataset. The 8^{th} row is insignificant with *mincard* value set to *2*, R′ = $\{8\}$.

$$R_1 = (R - R') = \{1, 2, 3, 4, 5, 6, 7\} \tag{3.2}$$

The equation 3.2 highlights the pruning of insignificant row $\{8\}$ from the bitTable shown in Table 3.2. Table 3.3 shows the bitTable after pruning the insignificant row $\{8\}$. The pruning of insignificant rows affects the support of features (*sup(*f_j*)*, $\forall\ f_j \in F_1$). The pruning of insignificant row $\{8\}$ will reduce the support of feature $\{e\}$ and eventually converts it to an insignificant feature, F′ = $\{e\}$. The preprocessing technique in the existing colossal itemset mining algorithms Nguyen *et al.* (2017*b*, 2016, 2017*a*) and FCCI mining algorithm Zulkurnain *et al.* (2012) prune the insignificant features and insignificant rows just once, and after that fails to take advantage of the reduction in the support of features due to the pruning of insignificant rows. Hence, the preprocessing technique used in the existing colossal itemset mining algorithms and FCCI mining algorithms fails to prune the complete set of insignificant features and insignificant rows. To overcome this drawback of existing preprocessing techniques, the proposed Effective Improved Preprocessing (EIP) technique takes advantage of the reduction in the support of features due to the pruning of insignificant rows and continues to prune the insignificant features.

$$F_2 = (F_1 - F') = \{a, b, d, f, g, h, i, j\} \tag{3.3}$$

The equation 3.3 highlights the pruning of insignificant feature $\{e\}$ from the bitTable as shown in Table 3.3. Table 3.4 shows the bitTable after pruning the insignificant feature $\{e\}$. Pruning the insignificant features affects the cardinality of rows (*card(rid)*, $\forall$ *rid* $\in R_1$). The pruning of insignificant feature $\{e\}$ will reduce the cardinality of the 7th row and eventually converting it to insignificant row, R′ = {*7*}. The proposed EIP technique takes advantage of the reduction in the cardinality of rows due to the pruning of insignificant features and continues to prune the insignificant rows. The proposed EIP technique prunes the insignificant features and insignificant rows alternatively in an iterative manner until all the features and rows in the bitTable satisfy the criteria of *minsup* and *mincard* respectively, whereas, the existing preprocessing technique prune the insignificant features and insignificant rows just once.

Table 3.5. bitTable after pruning insignificant row {*7*}

rid	a	b	d	f	g	h	i	j	***rs***
1	1	1	1	1	0	0	0	1	5
2	1	1	1	0	1	1	0	0	5
3	0	1	1	0	1	1	0	0	4
4	1	1	0	1	0	1	1	1	6
5	1	0	1	0	1	1	0	1	5
6	0	1	1	0	0	0	1	0	3
cs	4	5	5	2	3	4	2	3	

$$R_2 = (R_1 - R') = \{1, 2, 3, 4, 5, 6\} \tag{3.4}$$

The equation 3.4 highlights the pruning of insignificant row $\{7\}$ from the bitTable shown in Table 3.4. Table 3.5 shows the bitTable after pruning the insignificant row *7*. All the features and rows in the bitTable shown in Table 3.5 satisfy the criteria of *minsup* and *mincard*, hence the reduction terminates. Table 3.5 shows the bitTable after applying proposed EIP technique on high dimensional dataset shown in Table 1.1, with the *minsup* and *mincard* values set to *2*. Table 3.3 shows the bitTable after applying preprocessing technique of the existing algorithms on high dimensional dataset shown in Table 1.1, with the *minsup* and *mincard* values set to *2*. It is observed that proposed EIP technique prunes the complete set of insignificant features and insignificant rows, which is a limitation of the preprocessing technique in existing algorithms. Table 3.6 shows the bitTable after applying proposed EIP technique on high dimensional dataset shown in Table 1.1, with the *minsup* and *mincard* values set to *3*. After applying proposed EIP technique, let F_{final} be the set of significant features and R_{final} be the set of significant rows.

Table 3.6. Preprocessed bitTable when *minsup*=3 and *mincard*=3

rid	*a*	*b*	*d*	*g*	*h*	*j*	***rs***
1	1	1	1	0	0	1	4
2	1	1	1	1	1	0	5
3	0	1	1	1	1	0	4
4	1	1	0	0	1	1	4
5	1	0	1	1	1	1	5
cs	4	4	4	3	4	3	

3.2 Algorithm for Proposed Effective Improved Preprocessing Technique

The proposed EIP technique is divided into two tasks, Minimum Support Threshold Preprocessing (MSTP) task to prune the insignificant features and Minimum Cardinality Threshold Preprocessing (MCTP) task to prune the insignificant rows. Algorithm 1.1 shows the proposed EIP technique. Procedure 1.1a and Procedure 1.1b shows the MSTP task and MCTP task respectively. The proposed EIP technique invokes the MSTP and MCTP tasks in an iterative manner until all the features and rows in the bitTable satisfy

Algorithm 1.1. Proposed Effective Improved Preprocessing (EIP) Technique

```
Input: bitTable, minsup, mincard.
Output: preprocessed bitTable
    Initialisation : flag = 1
1:  while (flag==1) do
2:      MSTP_task()
3:      flag=0
4:      MCTP_task()
5:  end while
```

Procedure 1.1a. MSTP_task()

```
1:  for (j=0;j<no_of_features;) do
2:      count=0
3:      for (i=0;i<no_of_samples;i++) do
4:          if (bitTable[i][j]==1) then
5:              count++
6:          end if
7:      end for
8:      if (count<minsup) then
9:          delete $j^{th}$ feature
10:         no_of_features=no_of_features-1
11:     else
12:         j++
13:     end if
14: end for
```

Procedure 1.1b. MCTP_task()

```
1:  for (i=0;i<no_of_samples;) do
2:      count=0
3:      for (j=0;j<no_of_features;j++) do
4:          if (bitTable[i][j]==1) then
5:              count++
6:          end if
7:      end for
8:      if (count<mincard) then
9:          delete $i^{th}$ sample
10:         no_of_samples=no_of_samples-1
11:         flag=1
12:     else
13:         i++
14:     end if
15: end for
```

the criteria of *minsup* and *mincard* respectively.

3.3 Results and Discussion

This section demonstrates the effectiveness of the proposed EIP technique. The experiments have been conducted on ovarian cancer, lung cancer, prostate cancer, Mixed Lineage Leukemia (MLL), Central Nervous System embryonal tumor, Diffuse Large B-Cell Lymphoma (DLBCL) including Follicular Lymphoma, Lung Cancer Test, Acute Lymphoblastic Leukemia-Acute Myeloid Leukemia (ALL-AML), and Diffuse Large B-Cell Lymphoma (DLBCL) high dimensional datasets. The data characteristics of these high dimensional datasets are shown in Table 3.7. The proposed EIP technique has been compared with the preprocessing technique of CP-Miner, PCP-Miner, BVBUC, DisClose, and Pattern Fusion (PF). The experiments have been carried out on a computer with a specification of 3.4GHz core i7-3770 CPU, 8GB RAM, and 1TB hard disk.

For different values of *minsup* and *mincard*, the proposed EIP technique is compared with preprocessing technique of PCP-Miner, CP-Miner, BVBUC, DisClose and Pattern Fusion (PF) algorithms for ovarian cancer, lung cancer, prostate cancer, MLL,

Table 3.7. High Dimensional Biological-Datasets

Dataset	Number of rows (samples)	Number of features
Ovarian Cancer	253	15154
Lung Cancer	181	12533
Prostate Cancer	102	12600
Mixed Lineage Leukemia (MLL)	72	12582
Central Nervous System embryonal tumor	60	7129
Diffuse Large B-Cell Lymphoma (DLBCL) including Follicular Lymphoma	77	7129
Lung Cancer Test	32	12533
Acute Lymphoblastic Leukemia-Acute Myeloid Leukemia (ALL-AML)	38	7129
Diffuse Large B-Cell Lymphoma (DLBCL)	58	7129

Table 3.8. Comparison of Proposed EIP Technique and Preprocessing Technique used in DisClose, BVBUC, PF, CP-Miner and PCP-Miner for Ovarian Cancer dataset with *minsup* set to 20 and 30.

→ *minsup*		20				30			
↓ *mincard*		EIP	DisClose	BVBUC	CP-Miner	EIP	DisClose	BVBUC	CP-Miner
				PF	PCP-Miner			PF	PCP-Miner
2000	Rows	206	220	253	253	200	217	253	253
	Features	13523	13589	13589	13589	12304	12500	12500	12500
3000	Rows	174	192	253	253	166	185	253	253
	Features	13417	13589	13589	13589	11884	12500	12500	12500
4000	Rows	134	152	253	253	126	145	253	253
	Features	12906	13589	13589	13589	11194	12500	12500	12500
5000	Rows	103	125	253	253	0	99	253	253
	Features	12380	13589	13589	13589	0	12500	12500	12500
6000	Rows	0	69	253	253	0	60	253	253
	Features	0	13589	13589	13589	0	12500	12500	12500

Table 3.9. Comparison of Proposed EIP Technique and Preprocessing Technique used in DisClose, BVBUC, PF, CP-Miner and PCP-Miner for Ovarian Cancer dataset with *minsup* set to 40 and 50.

→ *minsup*		40				50			
↓ *mincard*		EIP	DisClose	BVBUC	CP-Miner	EIP	DisClose	BVBUC	CP-Miner
				PF	PCP-Miner			PF	PCP-Miner
2000	Rows	190	205	253	253	175	202	253	253
	Features	11079	11427	11427	11427	9818	10246	10246	10246
3000	Rows	155	169	253	253	134	160	253	253
	Features	10445	11427	11427	11427	7936	10246	10246	10246
4000	Rows	81	129	253	253	0	122	253	253
	Features	6468	11427	11427	11427	0	10246	10246	10246
5000	Rows	0	89	253	253	0	78	253	253
	Features	0	11427	11427	11427	0	10246	10246	10246

Table 3.10. Comparison of Proposed EIP Technique and Preprocessing Technique used in DisClose, BVBUC, PF, CP-Miner and PCP-Miner for Lung Cancer dataset with *minsup* set to 20 and 30.

$\rightarrow$ *minsup*		20				30			
$\downarrow$ *mincard*		EIP	DisClose	BVBUC	CP-Miner	EIP	DisClose	BVBUC	CP-Miner
				PF	PCP-Miner			PF	PCP-Miner
2000	Rows	160	180	181	181	154	179	181	181
	Features	7733	7752	7752	7752	6573	6594	6594	6594
3000	Rows	145	177	181	181	140	173	181	181
	Features	7692	7752	7752	7752	6543	6594	6594	6594
4000	Rows	0	136	181	181	0	74	181	181
	Features	0	7752	7752	7752	0	6594	6594	6594
5000	Rows	0	7	181	181	0	0	181	181
	Features	0	7752	7752	7752	0	6594	6594	6594

Table 3.11. Comparison of Proposed EIP Technique and Preprocessing Technique used in DisClose, BVBUC, PF, CP-Miner and PCP-Miner for Lung Cancer dataset with *minsup* set to 40 and 50.

$\rightarrow$ *minsup*		40				50			
$\downarrow$ *mincard*		EIP	DisClose	BVBUC	CP-Miner	EIP	DisClose	BVBUC	CP-Miner
				PF	PCP-Miner			PF	PCP-Miner
2000	Rows	150	177	181	181	145	176	181	181
	Features	5817	5834	5834	5834	5210	5225	5225	5225
3000	Rows	135	169	181	181	131	166	181	181
	Features	5616	5834	5834	5834	4980	5225	5225	5225
4000	Rows	0	0	181	181	0	0	181	181
	Features	0	5834	5834	5834	0	5225	5225	5225
5000	Rows	0	0	181	181	0	0	181	181
	Features	0	5834	5834	5834	0	5225	5225	5225

Table 3.12. Comparison of Proposed EIP Technique and Preprocessing Technique used in DisClose, BVBUC, PF, CP-Miner and PCP-Miner for Prostate Cancer dataset with *minsup* set to 20 and 30.

$\rightarrow$ *minsup*		20				30			
$\downarrow$ *mincard*		EIP	DisClose	BVBUC	CP-Miner	EIP	DisClose	BVBUC	CP-Miner
				PF	PCP-Miner			PF	PCP-Miner
2000	Rows	96	102	102	102	94	101	102	102
	Features	7063	7079	7079	7079	6321	6350	6350	6350
3000	Rows	90	100	102	102	86	95	102	102
	Features	6989	7079	7079	7079	6212	6350	6350	6350
4000	Rows	86	95	102	102	81	92	102	102
	Features	6629	7079	7079	7079	6100	6350	6350	6350
5000	Rows	68	80	102	102	65	78	102	102
	Features	6358	7079	7079	7079	5849	6350	6350	6350
6000	Rows	0	10	102	102	0	3	102	102
	Features	0	7079	7079	7079	0	6350	6350	6350

Table 3.13. Comparison of Proposed EIP Technique and Preprocessing Technique used in DisClose, BVBUC, PF, CP-Miner and PCP-Miner for Prostate Cancer dataset with *minsup* set to 40 and 50.

$\rightarrow$ *minsup*		40				50			
$\downarrow$ *mincard*		EIP	DisClose	BVBUC	CP-Miner	EIP	DisClose	BVBUC	CP-Miner
				PF	PCP-Miner			PF	PCP-Miner
2000	Rows	92	101	102	102	91	100	102	102
	Features	5863	5892	5892	5892	5480	5510	5510	5510
3000	Rows	83	92	102	102	79	89	102	102
	Features	5742	5892	5892	5892	5388	5510	5510	5510
4000	Rows	77	86	102	102	65	83	102	102
	Features	5684	5892	5892	5892	5294	5510	5510	5510
5000	Rows	0	67	102	102	0	61	102	102
	Features	0	5892	5892	5892	0	5510	5510	5510
6000	Rows	0	0	102	102	0	0	102	102
	Features	0	5892	5892	5892	0	5510	5510	5510

Central Nervous System embryonal tumor, DLBCL including Follicular Lymphoma, Lung Cancer Test, ALL-AML, and DLBCL high dimensional datasets, this comparison have been tabulated from Table 3.8 - Table 3.25. Table 3.8 - Table 3.25 highlight the final set of significant features and significant rows after preprocessing. From the experimental results, it can be inferred that the proposed EIP technique prunes all insignificant features and insignificant rows in all cases when compared to the existing preprocessing techniques.

The preprocessing technique of BVBUC algorithm and PF algorithm prunes the insignificant features just once; hence it has been observed from the experimental results that for a given dataset, *minsup* and different values of *mincard*, the number of insignificant features pruned by the preprocessing technique of BVBUC and PF algorithm will remain same. For example, the number of significant features after applying the preprocessing technique of BVBUC algorithm and PF algorithm for ovarian cancer dataset with *minsup* value of 20 and different values of *mincard* will be 13589, as shown in Table 3.8. This indicates that the number of insignificant features pruned by the preprocessing technique of BVBUC and PF algorithm will remain same. Table 3.8 and Table 3.9 highlight that similar results have been observed for *minsup* values of 30, 40 and 50. The number of significant features after applying the preprocessing technique of BVBUC algorithm and PF algorithm for lung cancer dataset with *minsup* value of 20 and different values of *mincard* will be 7752, as shown in Table 3.10. This indicates that the number of insignificant features pruned by the preprocessing technique of BVBUC and PF algorithm will remain same. Table 3.10 - Table 3.25 highlight that similar results have been observed for other experimental high dimensional datasets.

The experimental results show that, for a given dataset, *minsup* and different values of *mincard* the preprocessing technique of BVBUC algorithm and PF algorithm fails to prune insignificant rows. For example, the number of significant rows after applying the preprocessing technique of BVBUC algorithm and PF algorithm for ovarian cancer dataset with *minsup* value of 20 and different values of *mincard* will be 253, as shown in Table 3.8. This indicates that the preprocessing technique of BVBUC algorithm and PF algorithm fails to prune insignificant rows. Table 3.8 and Table 3.9 highlight that similar results have been observed for *minsup* values of 30, 40 and 50. The number of significant rows after applying the preprocessing technique of BVBUC algorithm and PF algorithm for lung cancer dataset with *minsup* value of 20 and different values of *mincard* will be 181, as shown in Table 3.10. This indicates that the preprocessing technique of BVBUC algorithm and PF algorithm fails to prune insignificant rows. Table 3.10 - Table 3.25 highlight that similar results have been observed for other experimental high dimensional datasets.

The preprocessing technique of PCP-Miner algorithm and CP-Miner algorithm prunes the insignificant features and then rows which do not contain any features, only once. The experimental results show that the number of rows pruned by the preprocessing technique of PCP-Miner and CP-Miner algorithm in all the experimental high dimensional datasets for a given *minsup* and *mincard* is zero. This illustrates that the number of rows in all experimental high dimensional datasets that do not contain any features is zero.

The number of insignificant features pruned by the preprocessing technique of PCP-Miner, CP-Miner and DisClose algorithm for a given dataset, *minsup* and different val-

Table 3.14. Comparison of Proposed EIP Technique and Preprocessing Technique used in DisClose, BVBUC, PF, CP-Miner and PCP-Miner for MLL dataset with *minsup* set to 5 and 10.

$\rightarrow$ *minsup*		5				10			
$\downarrow$ *mincard*		EIP	DisClose	BVBUC	CP-Miner	EIP	DisClose	BVBUC	CP-Miner
				PF	PCP-Miner			PF	PCP-Miner
2000	Rows	72	72	72	72	72	72	72	72
	Features	11144	11144	11144	11144	9360	9360	9360	9360
2500	Rows	72	72	72	72	72	72	72	72
	Features	11144	11144	11144	11144	9360	9360	9360	9360
3000	Rows	70	71	72	72	66	69	72	72
	Features	11133	11144	11144	11144	9142	9360	9360	9360
3500	Rows	60	65	72	72	55	62	72	72
	Features	10911	11144	11144	11144	8787	9360	9360	9360
4000	Rows	44	53	72	72	0	44	72	72
	Features	10499	11144	11144	11144	0	9360	9360	9360
4500	Rows	20	32	72	72	0	28	72	72
	Features	8339	11144	11144	11144	0	9360	9360	9360
5000	Rows	0	16	72	72	0	12	72	72
	Features	0	11144	11144	11144	0	9360	9360	9360

Table 3.15. Comparison of Proposed EIP Technique and Preprocessing Technique used in DisClose, BVBUC, PF, CP-Miner and PCP-Miner for MLL dataset with *minsup* set to 15 and 20.

$\rightarrow$ *minsup*		15				20			
$\downarrow$ *mincard*		EIP	DisClose	BVBUC	CP-Miner	EIP	DisClose	BVBUC	CP-Miner
				PF	PCP-Miner			PF	PCP-Miner
2000	Rows	72	72	72	72	72	72	72	72
	Features	7670	7670	7670	7670	6253	6253	6253	6253
2500	Rows	70	71	72	72	67	70	72	72
	Features	7583	7670	7670	7670	5868	6253	6253	6253
3000	Rows	64	68	72	72	59	65	72	72
	Features	7191	7670	7670	7670	5396	6253	6253	6253
3500	Rows	0	51	72	72	0	41	72	72
	Features	0	7670	7670	7670	0	6253	6253	6253
4000	Rows	0	28	72	72	0	15	72	72
	Features	0	7670	7670	7670	0	6253	6253	6253

Table 3.16. Comparison of Proposed EIP Technique and Preprocessing Technique used in DisClose, BVBUC, PF, CP-Miner and PCP-Miner for MLL dataset with *minsup* set to 25 and 30.

$\rightarrow$ *minsup*		25				30			
$\downarrow$ *mincard*		EIP	DisClose	BVBUC	CP-Miner	EIP	DisClose	BVBUC	CP-Miner
				PF	PCP-Miner			PF	PCP-Miner
1000	Rows	72	72	72	72	72	72	72	72
	Features	5066	5066	5066	5066	4186	4186	4186	4186
1500	Rows	70	71	72	72	68	70	72	72
	Features	4946	5066	5066	5066	4097	4186	4186	4186
2000	Rows	66	70	72	72	59	68	72	72
	Features	4817	5066	5066	5066	3806	4186	4186	4186
2500	Rows	60	64	72	72	0	60	72	72
	Features	4479	5066	5066	5066	0	4186	4186	4186
3000	Rows	0	57	72	72	0	29	72	72
	Features	0	5066	5066	5066	0	4186	4186	4186

Table 3.17. Comparison of Proposed EIP Technique and Preprocessing Technique used in DisClose, BVBUC, PF, CP-Miner and PCP-Miner for Central Nervous System embryonal tumor dataset with *minsup* set to 5 and 10.

$\rightarrow$ *minsup*		5				10			
$\downarrow$ *mincard*		EIP	DisClose	BVBUC	CP-Miner	EIP	DisClose	BVBUC	CP-Miner
				PF	PCP-Miner			PF	PCP-Miner
1000	Rows	60	60	60	60	60	60	60	60
	Features	6000	6000	6000	6000	4550	4550	4550	4550
1500	Rows	58	59	60	60	55	58	60	60
	Features	5932	6000	6000	6000	4471	4550	4550	4550
2000	Rows	52	55	60	60	50	55	60	60
	Features	5796	6000	6000	6000	4315	4550	4550	4550
2500	Rows	11	23	60	60	0	13	60	60
	Features	3709	6000	6000	6000	0	4550	4550	4550
3000	Rows	0	6	60	60	0	3	60	60
	Features	0	6000	6000	6000	0	4550	4500	4500

Table 3.18. Comparison of Proposed EIP Technique and Preprocessing Technique used in DisClose, BVBUC, PF, CP-Miner and PCP-Miner for Central Nervous System embryonal tumor dataset with *minsup* set to 15 and 20.

$\rightarrow$ *minsup*		15				20			
$\downarrow$ *mincard*		EIP	DisClose	BVBUC	CP-Miner	EIP	DisClose	BVBUC	CP-Miner
				PF	PCP-Miner			PF	PCP-Miner
500	Rows	60	60	60	60	60	60	60	60
	Features	3511	3511	3511	3511	2858	2858	2858	2858
1000	Rows	58	59	60	60	57	58	60	60
	Features	3481	3511	3511	3511	2766	2858	2858	2858
1500	Rows	52	56	60	60	50	54	60	60
	Features	3198	3511	3511	3511	2642	2858	2858	2858
2000	Rows	0	38	60	60	0	27	60	60
	Features	0	3511	3511	3511	0	2858	2858	2858

Table 3.19. Comparison of Proposed EIP Technique and Preprocessing Technique used in DisClose, BVBUC, PF, CP-Miner and PCP-Miner for Central Nervous System embryonal tumor dataset with *minsup* set to 25 and 30.

→ *minsup*		25				30			
↓ *mincard*		EIP	DisClose	BVBUC	CP-Miner	EIP	DisClose	BVBUC	CP-Miner
				PF	PCP-Miner			PF	PCP-Miner
500	Rows	60	60	60	60	60	60	60	60
	Features	2455	2455	2455	2455	2144	2144	2144	2144
1000	Rows	55	57	60	60	52	55	60	60
	Features	2295	2455	2455	2455	2016	2144	2144	2144
1500	Rows	0	50	60	60	0	47	60	60
	Features	0	2455	2455	2455	0	2144	2144	2144
2000	Rows	0	12	60	60	0	0	60	60
	Features	0	2455	2455	2455	0	2144	2144	2144

Table 3.20. Comparison of Proposed EIP Technique and Preprocessing Technique used in DisClose, BVBUC, PF, CP-Miner and PCP-Miner for DLBCL (Including Follicular Lymphoma) dataset with *minsup* set to 5 and 10.

→ *minsup*		5				10			
↓ *mincard*		EIP	DisClose	BVBUC	CP-Miner	EIP	DisClose	BVBUC	CP-Miner
				PF	PCP-Miner			PF	PCP-Miner
1000	Rows	77	77	77	77	77	77	77	77
	Features	5904	5904	5904	5904	4751	4751	4751	4751
1500	Rows	75	76	77	77	74	76	77	77
	Features	5826	5904	5904	5904	4616	4751	4751	4751
2000	Rows	74	75	77	77	73	75	77	77
	Features	5786	5904	5904	5904	4542	4751	4751	4751
2500	Rows	58	65	77	77	48	57	77	77
	Features	5609	5904	5904	5904	4154	4751	4751	4751
3000	Rows	0	17	77	77	0	11	77	77
	Features	0	5904	5904	5904	0	4751	4751	4751

Table 3.21. Comparison of Proposed EIP Technique and Preprocessing Technique used in DisClose, BVBUC, PF, CP-Miner and PCP-Miner for DLBCL (Including Follicular Lymphoma) dataset with *minsup* set to 15 and 20.

$\rightarrow$ *minsup*		15				20			
$\downarrow$ *mincard*		EIP	DisClose	BVBUC	CP-Miner	EIP	DisClose	BVBUC	CP-Miner
				PF	PCP-Miner			PF	PCP-Miner
1000	Rows	77	77	77	77	75	76	77	77
	Features	3928	3928	3928	3928	3342	3409	3409	3409
1500	Rows	73	75	77	77	71	74	77	77
	Features	3852	3928	3928	3928	3192	3409	3409	3409
2000	Rows	71	73	77	77	68	71	77	77
	Features	3755	3928	3928	3928	3015	3409	3409	3409
2500	Rows	0	44	77	77	0	38	77	77
	Features	0	3928	3928	3928	0	3409	3409	3409

Table 3.22. Comparison of Proposed EIP Technique and Preprocessing Technique used in DisClose, BVBUC, PF, CP-Miner and PCP-Miner for DLBCL (Including Follicular Lymphoma) dataset with *minsup* set to 25 and 30.

$\rightarrow$ *minsup*		25				30			
$\downarrow$ *mincard*		EIP	DisClose	BVBUC	CP-Miner	EIP	DisClose	BVBUC	CP-Miner
				PF	PCP-Miner			PF	PCP-Miner
1000	Rows	74	76	77	77	72	75	77	77
	Features	2993	3066	3066	3066	2714	2822	2822	2822
1500	Rows	70	73	77	77	67	71	77	77
	Features	2847	3066	3066	3066	2597	2822	2822	2822
2000	Rows	65	69	77	77	61	67	77	77
	Features	2722	3066	3066	3066	2425	2822	2822	2822
2500	Rows	0	26	77	77	0	6	77	77
	Features	0	3066	3066	3066	0	2822	2822	2822

ues of *mincard* will remain the same. For example, the number of significant features after applying the preprocessing technique of PCP-Miner, CP-Miner and DisClose for ovarian cancer dataset with *minsup* value of 20 and different values of *mincard* will be 13589, as shown in Table 3.8. This indicates that the number of insignificant features pruned by the preprocessing technique of BVBUC and PF algorithm will remain same. Table 3.8 and Table 3.9 highlight that similar results have been observed for *minsup* values of 30, 40 and 50. The number of significant features after applying the preprocessing technique of PCP-Miner, CP-Miner and DisClose for lung cancer dataset with *minsup* value of 20 and different values of *mincard* will be 7752, as shown in Table 3.10. This indicates that the number of insignificant features pruned by the preprocessing technique of BVBUC and PF algorithm will remain same. Table 3.10 to Table 3.25 highlight that similar results have been observed for other experimental high dimensional datasets. The pruning of insignificant features remains the same because the preprocessing technique of PCP-Miner, CP-Miner and DisClose algorithm prune the insignificant features just once and fail to take advantage of the reduction in the support of features due to the pruning of insignificant rows.

The experimental results highlight that the proposed EIP technique takes advantage of the reduction in the cardinality of rows due to the pruning of insignificant features and the reduction in the support of features due to the pruning of insignificant rows.

Table 3.23. Comparison of Proposed EIP Technique and Preprocessing Technique used in DisClose, BVBUC, PF, CP-Miner and PCP-Miner for Lung Cancer Test dataset with *minsup* set to 10 and 15.

$\rightarrow$ *minsup*		10				15			
$\downarrow$ *mincard*		EIP	DisClose	BVBUC	CP-Miner	EIP	DisClose	BVBUC	CP-Miner
				PF	PCP-Miner			PF	PCP-Miner
500	Rows	32	32	32	32	32	32	32	32
	Features	6825	6825	6825	6825	4873	4873	4873	4873
1000	Rows	32	32	32	32	32	32	32	32
	Features	6825	6825	6825	6825	4873	4873	4873	4873
1500	Rows	32	32	32	32	32	32	32	32
	Features	6825	6825	6825	6825	4873	4873	4873	4873
2000	Rows	31	31	32	32	30	31	32	32
	Features	6810	6825	6825	6825	4727	4873	4873	4873

Table 3.24. Comparison of Proposed EIP Technique and Preprocessing Technique used in DisClose, BVBUC, PF, CP-Miner and PCP-Miner for ALL-AML dataset with *minsup* set to 10 and 15.

$\rightarrow$ *minsup*		10				15			
$\downarrow$ *mincard*		EIP	DisClose	BVBUC	CP-Miner	EIP	DisClose	BVBUC	CP-Miner
				PF	PCP-Miner			PF	PCP-Miner
1000	Rows	38	38	38	38	36	36	38	38
	Features	4179	4179	4179	4179	3265	3265	3265	3265
2000	Rows	20	26	38	38	20	22	38	38
	Features	3233	4179	4179	4179	3233	3265	3265	3265
3000	Rows	20	22	38	38	19	20	38	38
	Features	3233	4179	4179	4179	3212	3265	3265	3265
4000	Rows	0	0	38	38	0	0	38	38
	Features	0	4179	4179	4179	0	3265	3265	3265

Table 3.25. Comparison of Proposed EIP Technique and Preprocessing Technique used in DisClose, BVBUC, PF, CP-Miner and PCP-Miner for DLBCL dataset with *minsup* set to 15 and 20.

$\rightarrow$ *minsup*		15				20			
$\downarrow$ *mincard*		EIP	DisClose	BVBUC	CP-Miner	EIP	DisClose	BVBUC	CP-Miner
				PF	PCP-Miner			PF	PCP-Miner
500	Rows	58	58	58	58	58	58	58	58
	Features	3572	3572	3572	3572	2898	2898	2898	2898
1000	Rows	58	58	58	58	58	58	58	58
	Features	3572	3572	3572	3572	2898	2898	2898	2898
1500	Rows	55	57	58	58	52	55	58	58
	Features	3505	3572	3572	3572	2683	2898	2898	2898
2000	Rows	0	40	58	58	0	30	58	58
	Features	0	3572	3572	3572	0	2898	2898	2898

The proposed EIP technique prunes the insignificant features, and insignificant rows alternatively in an iterative manner until all the features and rows satisfy the criteria of *minsup* and *mincard* respectively. Further, the number of significant features and significant rows after the proposed EIP technique is zero, when the (*minsup*, *mincard*) values reach (20, 6000), (30, 5000), (40, 5000), (50, 4000) for the ovarian cancer dataset as shown in Table 3.8 and Table 3.9. This indicates that there are no frequent colossal itemsets and FCCI. In such cases, the proposed EIP technique yields the final results without the need for the proposed frequent colossal itemset mining algorithm and FCCI mining algorithm. However, the existing FCCI mining algorithms have to go through an enormous number of row combinations to fetch the same final results even after applying their respective preprocessing techniques.

The number of significant features and significant rows after the proposed EIP technique is zero, when the (*minsup*, *mincard*) values reach (5, 5000), (10, 4000), (15, 3500), (20, 3500), (25, 3000), (30, 2500) for the MLL dataset as shown in Table 3.14, Table 3.15, and Table 3.16. This indicates that there are no frequent colossal itemsets and FCCI. In such cases, the proposed EIP technique yields the final results without the need for the proposed frequent colossal itemset mining algorithm and FCCI mining algorithm. However, the existing FCCI mining algorithms have to go through an enormous number of row combinations to fetch the same final results even after applying their respective preprocessing techniques. Similarly, the same kind of results have been observed when the proposed EIP technique is compared with existing preprocessing techniques for the lung cancer dataset (as shown in Table 3.10 and Table 3.11), prostate cancer dataset (as shown in Table 3.12 and Table 3.13), Central Nervous System embryonal tumor dataset (as shown in Table 3.17, Table 3.18 and Table 3.19), DLBCL including Follicular Lymphoma dataset (as shown in Table 3.20, Table 3.21 and Table 3.22), lung cancer test dataset (as shown in Table 3.23), ALL-AML dataset (as shown in Table 3.24, and DLBCL dataset (as shown in Table 3.25). The proposed EIP technique outperforms the existing preprocessing techniques effectively by pruning the complete set of insignificant features and insignificant rows. Further it has been observed for all the experimental high dimensional datasets that as the *minsup* and *mincard* increases the number of significant features and significant rows decreases.

3.4 Summary

In this chapter, the Effective Improved Preprocessing (EIP) technique has been proposed to prune the complete set of insignificant features and insignificant rows from the high dimensional dataset. The preprocessing technique of the existing frequent colossal itemset mining algorithms and FCCI mining algorithms do not prune the complete set of insignificant features and insignificant rows from the high dimensional dataset. The proposed EIP effectively utilizes *minsup* and *mincard* to prune the complete set of insignificant features and insignificant rows from the high dimensional dataset. The proposed EIP technique takes advantage of the reduction in the cardinality of rows due to the pruning of insignificant features and the reduction in the support of features due to the pruning of insignificant rows. The experiments have been conducted on high dimensional datasets for different values of *minsup* and *mincard*. It is evident from the experimental results that the proposed EIP technique outperforms the existing pre-processing technique in terms of pruning the complete set of insignificant features and insignificant rows, which further helps in reducing the search space.

In the next chapter, the proposed frequent colossal itemset mining and FCCI mining algorithm have been discussed.

Chapter 4

Mining Frequent Colossal Itemsets and Frequent Colossal Closed Itemsets from the High Dimensional Dataset

In this chapter, the proposed frequent colossal itemset mining algorithm has been discussed. The chapter also discusses the proposed algorithm enclosed with an efficient Rowset Cardinality Table (RCT) based closeness checking method and pruning strategy, which efficiently mine FCCI from the high dimensional dataset.

4.1 Search Strategies

Frequent colossal itemsets and FCCI are efficiently mined from the high dimensional dataset by choosing an efficient search strategy to traverse an enumerated tree. Bottom-up or top-down search strategy is used to traverse an enumerated tree. An enumerated tree can either be a row enumerated tree or a feature enumerated tree.

The proposed work mine frequent colossal itemsets and FCCI from the high dimensional dataset by traversing a row enumerated tree due to the data characteristics of the high dimensional datasets. In this section, the reason for the selection of a bottom-up search strategy to traverse the row enumerated tree has been explained.

4.1.1 Top-Down Traversal of Row Enumerated Tree

The top-down traversing of the row enumerated space implies that the search starts from the larger rowset value and builds the smaller rowset values during the process. Figure 4.1 shows the top-down row enumerated tree for the preprocessed bitTable 3.5, with each row enumerated node representing the rowset and the corresponding bitset result is indicated under the corresponding row enumerated node. The bitset at each row enumerated node helps in determining the cardinality of an itemset and is obtained by performing bitwise AND operations of bitset that corresponds to the *rid*s in the rowset. For example, the bitset at row enumerated node *123* (01100000) shown in Figure 4.1 is obtained by performing the bitwise AND operations of bitset that corresponds to the *rid*s 1 (11110001), 2 (11101100) and 3 (01101100). The top-down traversal of the row enumerated tree is inefficient in mining colossal itemsets as the large cardinality itemsets are present at the final levels of top-down row enumerated tree. The top-down traversal of the row enumerated tree expends a huge amount of time in traversing small and mid-sized itemsets at the initial levels of the row enumerated tree.

The top-down traversal of the row enumerated tree is inefficient with bitset approach as the bitset result at a parent row enumerated node cannot be utilized to generate bitset result at the child row enumerated node. The number of bitwise AND operations required to obtain the bitset result at the row enumerated node is (|rowset| - 1). For example, in Figure 4.1, the bitset result at node *1236* requires 3 bitwise AND operations. The top-down approach fails to take advantage of the anti-monotone property of minimum cardinality threshold. These disadvantages of top-down traversal of row enumerated tree make it as an inefficient search strategy for mining frequent colossal itemsets and FCCI from the high dimensional dataset.

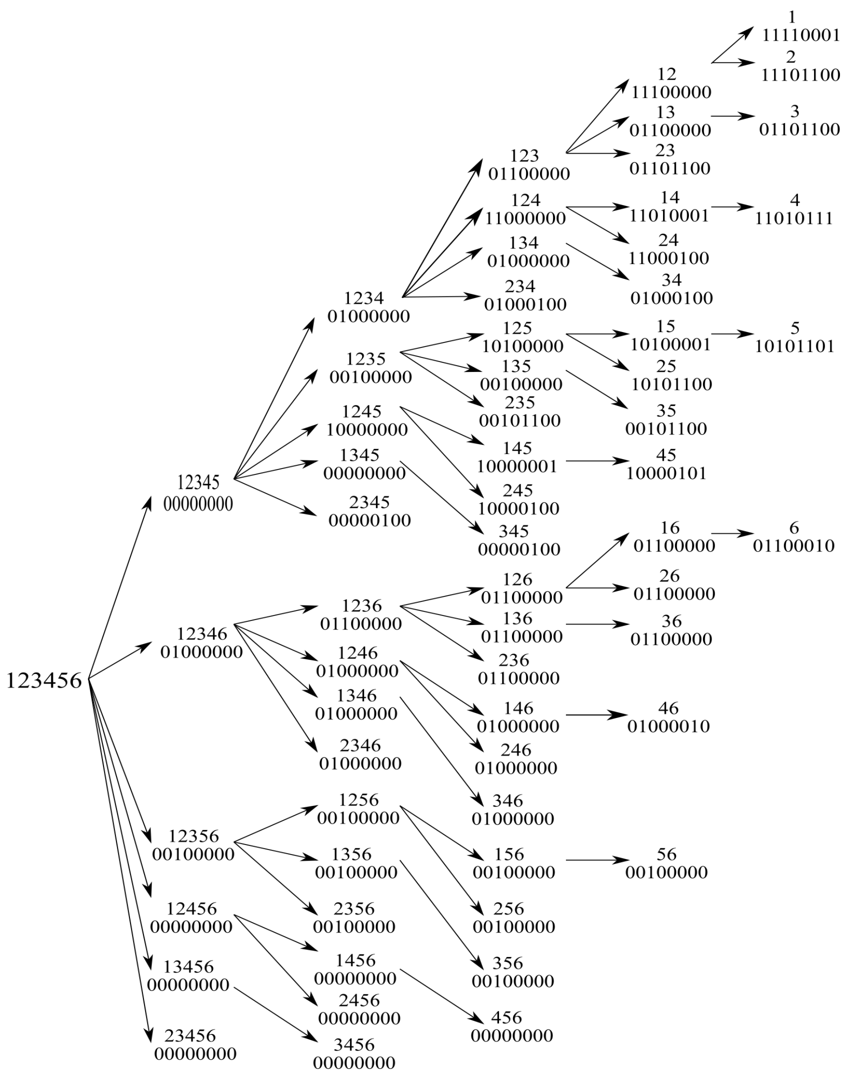


Figure 4.1. Top-Down Traversal of Row Enumerated Tree

4.1.2 Bottom-Up Traversal of Row Enumerated Tree

The bottom-up traversing of the row enumerated space implies that the search starts from the smaller rowset value and builds the larger rowset values during the process. Figure 4.2 shows the bottom-up row enumerated tree for the preprocessed bitTable 3.5, with each row enumerated node representing the rowset and the corresponding bitset result is indicated under the corresponding row enumerated node.

The bottom-up traversal of the row enumerated tree is efficient in mining colossal itemsets as the large cardinality itemsets are present at the initial levels of bottom-up row enumerated tree. The bottom-up approach also has a benefit of utilizing the bitset result of the parent row enumerated node to obtain the bitset result at the child row enumerated nodes, thus exponentially reducing the number of bitwise AND operations

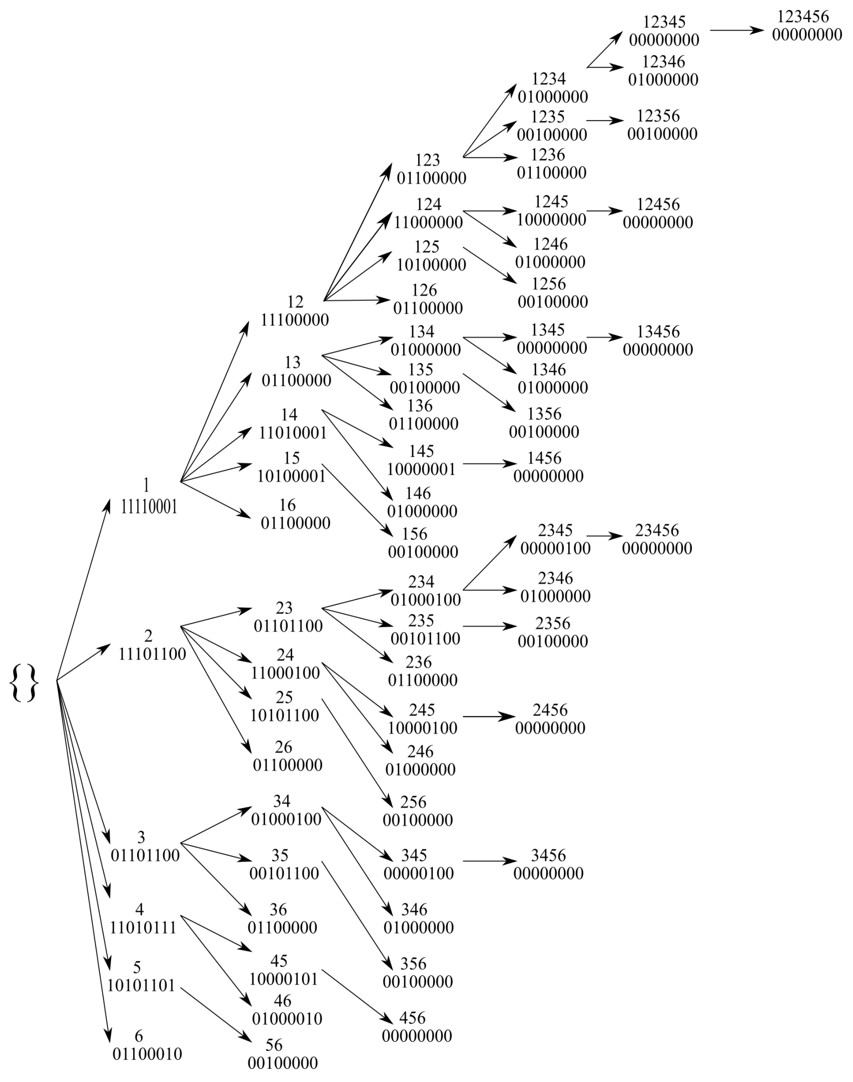


Figure 4.2. Bottom-Up Traversal of Row Enumerated Tree

to be performed. Only one bitwise AND operation is required to obtain the bitset result at each node in bottom-up row enumerated tree, as compared to (|rowset| - 1) in top-down row enumerated tree. For example, in Figure 4.2, the bitset result at node *1236* requires one bitwise AND operation. The bitset result of node *123* and node *6* are utilized to generate the bitset result of node *1236*.

The bottom-up approach efficiently take advantage of the anti-monotone property of minimum cardinality threshold to cut down the row enumerated search space. It means that, if an itemset at a node represented by *l*-rowset is not colossal, then an itemset at a child row enumerated node represented by (*l*+1)-rowset is also not colossal. For example, with *minsup* value set to 2 and *mincard* value set to 3, the descendants of row enumerated node *123* can be pruned as the itemsets in descendant row enumerated nodes are not colossal. Both bottom-up and top-down traversal of row enumerated tree have advantages and disadvantages. The proposed methods adopt the bottom-up traversal of row enumerated tree for mining frequent colossal itemsets and FCCI from the high dimensional dataset.

4.2 Proposed Frequent Colossal Itemset Mining from the High Dimensional Dataset

The existing frequent colossal itemset mining algorithms mine limited set of frequent colossal itemsets from the high dimensional dataset leading to the generation of an incomplete set of association rules. Frequent colossal itemset mining algorithm has been proposed to achieve better accuracy than existing algorithms in terms of mining number of frequent colossal itemsets from the high dimensional dataset.

The proposed frequent colossal itemset mining algorithm for the high dimensional dataset adopts the bottom-up traversal of row enumerated tree. The BVBUC algorithm mine the itemsets from the nodes belonging to the *minsup* level of row enumerated tree and prune their descendant row enumerated nodes. The proposed frequent colossal itemset mining algorithm mines the itemsets from the nodes belonging to the *minsup* level of row enumerated tree and from its descendant row enumerated nodes which contribute to the mining of frequent colossal itemsets.

The algorithm 4.1 highlights the proposed frequent colossal itemset mining algorithm. The proposed algorithm mine the frequent colossal itemsets by performing the depth-first traversal of bottom-up row enumerated tree. The preprocessed bitTable, minimum support threshold (*minsup*), and minimum cardinality threshold (*mincard*) are

Algorithm 4.1. Proposed Frequent Colossal Itemset Mining Algorithm

```
Input: preprocessed bitTable, minsup, mincard.
Output: frequent colossal itemsets
    Initialisation: frequent colossal itemsets = $\emptyset$
                    r = initial row in row enumeration
                    All bits in bitset_result initialized to 1
1: Colossal_Itemsets(r,bitset_result)
```

Procedure 4.1a. Colossal_Itemsets(*rcomb*,bitset_result)

```
1: if node rcomb does not reach till minsup
2:     return
3: calculate bitset_result at node rcomb
4: if ($|$bit_result$| <$ mincard)
5:     return
6: if ($|$rcomb$| \geq$ minsup)
7:     Add bitset_result to frequent colossal itemsets
8: for each node rcomb in row enumeration
9:     Colossal_Itemsets(rcomb,bitset_result)
```

provided as an input to the proposed frequent colossal itemset mining algorithm. The proposed algorithm provides set of frequent colossal itemsets as an output. The set of frequent colossal itemsets is initialized to null. Let '*r*' be the initial row in row enumeration and all the bits in bitset result are initialized to 1. The step 1 of procedure 4.1a indicates that, if the row enumerated node *rcomb* does not reach till *minsup* then, that row enumerated node and its subtree if exists are pruned to cut down the search space. For example, the row enumerated node *4* and *5* along with its subtree in Figure 4.2 are pruned when the *minsup* value is set to 4. The bit result is calculated at row enumerated node *rcomb*. The step 4 of procedure 4.1a indicates that, if the length of an itemset mined at row enumerated node *rcomb* is less than *mincard* then, the subtree of the row enumerated node *rcomb* is pruned to cut down the search space. The mined itemset is added to the set of frequent colossal itemsets if it is frequent and colossal. The algorithm continues with depth-first traversal of bottom-up row enumerated tree. The proposed algorithm does not stop mining colossal itemsets at the *minsup* level of row enumerated tree but continues mining colossal itemsets from higher levels of row enumerated tree which contributes to the mining of frequent colossal itemsets. The proposed algorithm achieves better accuracy than existing algorithms in terms of mining number of frequent colossal itemsets.

4.3 Proposed Frequent Colossal Closed Itemset Mining using Prune Table

The existing algorithms are inefficient in mining the FCCI from the high dimensional dataset. The pruning strategy of the existing frequent colossal closed itemset mining algorithms is inefficient. To overcome the drawback of existing algorithms, the frequent colossal closed itemset mining algorithm enclosed with an efficient pruning strategy has been proposed. The proposed frequent colossal closed itemset mining algorithm adopts the bottom-up row enumerated tree and bitset approach for mining FCCI from the high dimensional dataset. The proposed frequent colossal closed itemset mining algorithm has been enclosed with a pruning strategy to efficiently cut down the row enumerated search space. The pruning strategy takes advantage of the Prune Table (PT) to efficiently cut down the row enumerated search space.

Definition 9 (Prune Table). *Given a rowset Y=$\{r_{i1}$, r_{i2},......, $r_{ik}\}$ representing a row enumerated node in an order such that r_{i1}<r_{i2}<.....<r_{ik}, the Prune Table (PT_Y) contains the cardinality for all rids' of the preprocessed bitTable which are greater than the largest rid in Y i.e. $\forall$ $r_i \in R_{final}$, $r_i > r_{ik}$ at that particular row enumerated node*

Example 9. *Table 4.1a and 4.1b shows the Prune Table at row enumerated node 13 and 34 respectively in the Figure 4.2.*

Table 4.1. Prune Table (PT) of row enumerated node *13* (PT_{13}) and *34* (PT_{34})

(a) PT_{13}

rid	card
4	1
5	1
6	2

(b) PT_{34}

rid	card
5	1
6	1

The Prune Table PT_Y is obtained by the following steps.

1. Obtain the indices of all the ones appearing in the bitset result at that particular row enumerated node
2. For all *rid*s' of the preprocessed bitTable which are greater than the largest *rid* in *Y*, calculate the number of ones from the preprocessed bitTable appearing at the indices obtained from the step 1.

Example 10. *Obtaining Prune Table for row enumerated nodes 13 (PT_{13}) and 34 (PT_{34}) as shown in Figure 4.2.*

- *Prune Table PT_{13} is shown in Table 4.1a, the indices of all the ones appearing in the bitset result (01100000) at row enumerated node 13 are $\{$2,3$\}$.*
- *For all rids' $\{$4,5,6$\}$, which are greater than the largest rid $\{$3$\}$ at row enumerated node 13, the number of ones appearing at the indices $\{$2,3$\}$ from the preprocessed bitTable as shown in Table 3.5 are $\{$1,1,2$\}$ respectively.*
- *Prune Table PT_{34} is shown in Table 4.1b, the indices of all the ones appearing in the bitset result (01000100) at row enumerated node 34 are $\{$2,6$\}$.*
- *For all rids' $\{$5,6$\}$, which are greater than the largest rid $\{$4$\}$ at row enumerated node 34, the number of ones appearing at the indices $\{$2,6$\}$ from the preprocessed bitTable as shown in Table 3.5 are $\{$1,1$\}$ respectively.*

The Prune Table provides the prior information regarding the cardinality of the itemsets to be mined at the immediate child nodes without traversing them. For example, with *minsup* and *mincard* values set to 2, the Prune Table at row enumerated node *13* (PT_{13}) gives the prior information regarding the cardinality of itemsets to be mined at immediate child row enumerated nodes (*134*, *135*, *136*). The cardinality of *rid 4* and *rid 5* in PT_{13} does not satisfy the *mincard*, which indicates the proposed algorithm to prune the descendants of row enumerated node *13* related to *rid 4* and *rid 5* without traversing the descendants. The use of Prune Table provides a computational boost to the proposed frequent colossal closed itemset mining algorithm as it provides the prior information regarding the cardinality of the itemsets to be mined, which is a major limitation in the existing algorithms.

The algorithm 4.2 shows the proposed frequent colossal closed itemset mining algorithm. The proposed algorithm mine the FCCI by performing the depth-first traversal of bottom-up row enumerated tree. The preprocessed bitTable, minimum support threshold, and minimum cardinality threshold are provided as an input to the proposed algo-

Algorithm 4.2. Proposed Frequent Colossal Closed Itemset Mining algorithm

```
Input: preprocessed bitTable, minsup, mincard.
Output: Set of Frequent Colossal Closed Itemsets, SFCCI
    Initialisation: SFCCI= $\emptyset$
                $R'_{final}$ = set of rows to be enumerated
                rcomb = initial node in row enumeration
                All bits' in bitset_result are initialized to 1
1:     Colossal_Closed(rcomb,bitset_result,$R'_{final}$,SFCCI)
```

Procedure 4.2a. Colossal_Closed(*rcomb*,bitset_result,R'_{final},SFCCI))

```
1:  if node rcomb does not reach till minsup
2:      return
3:  calculate bitset_result at the node rcomb
4:  Pruning: $\forall\ r_i \in R'_{final}$, if $PT_{rcomb}[r_i]$[card]<mincard
5:                          delete $r_i$ from $R'_{final}$
6:  checking the closeness of an itemsets, if closenesscheck(itemset)==True
7:      Add itemset to SFCCI
8:     for each row enumerated node(rcomb) in $R'_{final}$
9:         Colossal_Closed(rcomb,bitset_result,$R'_{final}$,SFCCI)
```

rithm. The proposed algorithm provides a set of frequent colossal closed itemsets as an output. The SFCCI, Set of Frequent Colossal Closed Itemsets is initialized to null. Let R'_{final} be the set of rows to be enumerated, *rcomb* be the initial row enumerated node, and all the bits in bitset_result be initialized to 1.

The 'Colossal_Closed' procedure as shown in procedure 4.2a is invoked to mine the FCCI from the preprocessed bitTable. The step 1 in 'Colossal_Closed' procedure indicates that, if the row enumerated node *rcomb* does not reach till *minsup* then, that row enumerated node and its subtree if exists is pruned to cut down the search space. The step 3 highlights about obtaining the bitset_result at the row enuemrated node *rcomb*. Step 4 indicates the proposed PT based pruning strategy. R'_{final} indicates the number of rows to be enumerated at a node *rcomb*, and if the cardinality of any of the rows in R'_{final} occurring in PT_{rcomb} is less than *mincard* then those rows are removed from R'_{final} as they would not generate colossal itemsets. For example, *rid 4* and *rid 5* are removed from R'_{final} at row enumerated node *13* because the cardinality of *rid 4* and *rid 5* in PT_{13} as shown in Table 4.1a is less than *mincard*(2). Removal of *rid 4* and *rid 5* from R'_{final} leads to the pruning of row enumerated nodes *134*, *1345*, *13456*, *1346*, *135* and *1356* shown in Figure 4.2. The proposed pruning strategy utilizes the Prune Table (PT) as it provides the prior information regarding the cardinality of the itemsets to be mined at the immediate child nodes without traversing them. Step 6 highlights the existing closeness checking of itemset obtained at row enumerated node *rcomb*. If the itemset satisfies the closeness checking then the itemset is added to the SFCCI. The algorithm continues with depth-first traversal of bottom-up row enumerated tree.

4.4 Proposed Method for Mining Frequent Colossal Closed Itemsets from the High Dimensional Dataset using Rowset Cardinality Table

The existing algorithms are inefficient in mining the FCCI from the high dimensional dataset. The closeness checking method of rowset and pruning strategy of the existing frequent colossal closed itemset mining algorithms are inefficient. To overcome the drawback of existing algorithms, the following approaches have been proposed:

- Rowset Cardinality Table (RCT).
- An efficient RCT based closeness checking method to check the closeness of a rowset.
- An efficient RCT based pruning strategy to cut down the search space by efficient utilization of minimum cardinality threshold.
- The BitSet Frequent Colossal Closed Itemset Mining (BSFCCIM) algorithm entrenched with the efficient RCT based closeness checking method and pruning strategy has been proposed to efficiently mine FCCI from the high dimensional dataset.

The proposed Rowset Cardinality Table (RCT) is generated at a row enumerated node. The RCT based closeness checking method has been proposed to check whether a rowset is closed or not. The proposed RCT based closeness checking method will not scan through the previously mined FCCI to check the existence and closeness of newly mined frequent colossal itemset. It is not required to store the complete set of previously mined FCCI in main memory as the closeness of rowset indicates the closeness of itemsets mined at that particular rowset. The proposed pruning strategy utilizes the RCT at the row enumerated node to efficiently cut down the search space. For mining FCCI from the high dimensional dataset, the proposed algorithm utilizes the bottom-up row enumerated tree, as the large cardinality itemsets are present at the initial level of the bottom-up row enumerated tree. The bitset approach is computationally fast, hence the proposed algorithm utilizes the bitset approach. The proposed BSFCCIM algorithm is entrenched with efficient RCT based closeness checking method and pruning strategy.

4.4.1 Rowset Cardinality Table

The proposed closeness checking method takes advantage of the Rowset Cardinality Table (RCT) in bottom-up row enumerated tree to check the closeness of a rowset. If the rowset is closed then the itemset mined at that rowset is also closed. The RCT helps in closeness checking of a rowset without the need to scan through the previously mined FCCI itemsets. The proposed pruning strategy utilizes the RCT to efficiently cut down the row enumerated search space. The RCT provides the prior information regarding the cardinality of the itemsets to be mined at the descendant nodes without traversing them. The RCT for every row enumerated node is shown in Figure 4.3. Each row enumerated node refers their respective proposed Rowset Cardinality Table as shown in Figure 4.3 for closeness checking of a rowset and pruning the descendant row enumerated nodes which do not contribute to the mining of colossal itemsets.

Definition 10 (Rowset Cardinality Table)**.** *Given a rowset Y, representing a row enumerated node, the Rowset Cardinality Table (RCT_Y) at row enumerated node Y contains the updated cardinality for each row in (R_{final} - Y) depending upon the cardinality of the bitset result obtained at row enumerated node Y.*

The Rowset Cardinality Table (RCT_Y) at node Y is obtained by the following steps.

1. Obtain the indices of all the ones appearing in the bitset result obtained at row enumerated node Y.
2. For each row in (R_{final} - Y), calculate the number of ones from the preprocessed bitTable appearing at the indices obtained from the step 1.

Example 11. *Table 4.2a, 4.2b, 4.2c and 4.2d shows the Rowset Cardinality Table at row enumerated nodes 12, 13, 14 and 23 respectively in the Figure 4.3*

Table 4.2. Rowset Cardinality Table of row enumerated nodes *12*, *13*, *14* and *23*

(a) RCT_{12}

rid	card
3	2
4	2
5	2
6	2

(b) RCT_{13}

rid	card
2	2
4	1
5	1
6	2

(c) RCT_{14}

rid	card
2	2
3	1
5	2
6	1

(d) RCT_{23}

rid	card
1	2
4	2
5	3
6	2

Example 12. *Obtaining Rowset Cardinality Table for row enumerated nodes 14 (RCT_{14}) and 23 (RCT_{23}) as shown in Figure 4.3.*

- *Rowset Cardinality Table for row enumerated node 14, RCT_{14} is shown in Table 4.2c. The indices of all the ones appearing in the bitset result (11010001) at row enumerated node 14 are {1,2,4,8}.*

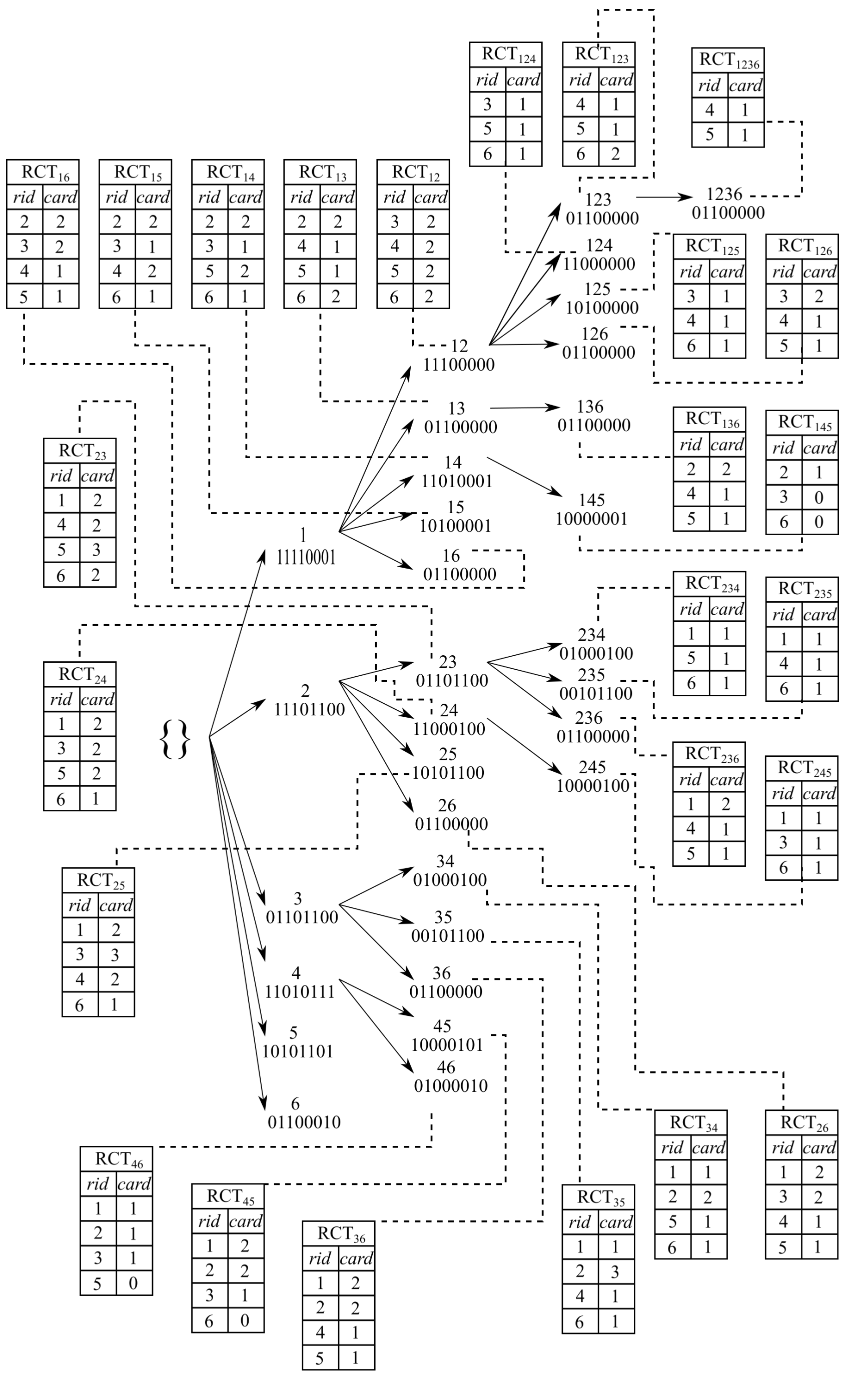

{}
1 11110001
2 11101100
3 01101100
4 11010111
5 10101101
6 01100010
12 11100000
13 01100000
14 11010001
15 10100001
16 01100000
23 01101100
24 11000100
25 10101100
26 01100000
34 01000100
35 00101100
36 01100000
45 10000101
46 01000010
123 01100000
124 11000000
125 10100000
126 01100000
136 01100000
145 10000001
234 01000100
235 00101100
236 01100000
245 10000100
1236 01100000
RCT124 rid card 3 1 5 1 6 1
RCT123 rid card 4 1 5 1 6 2
RCT1236 rid card 4 1 5 1
RCT16 rid card 2 2 3 2 4 1 5 1
RCT15 rid card 2 2 3 1 4 2 6 1
RCT14 rid card 2 2 3 1 5 2 6 1
RCT13 rid card 2 2 4 1 5 1 6 2
RCT12 rid card 3 2 4 2 5 2 6 2
RCT125 rid card 3 1 4 1 6 1
RCT126 rid card 3 2 4 1 5 1
RCT136 rid card 2 2 4 1 5 1
RCT145 rid card 2 1 3 0 6 0
RCT23 rid card 1 2 4 2 5 3 6 2
RCT234 rid card 1 1 5 1 6 1
RCT235 rid card 1 1 4 1 6 1
RCT24 rid card 1 2 3 2 5 2 6 1
RCT236 rid card 1 2 4 1 5 1
RCT245 rid card 1 1 3 1 6 1
RCT25 rid card 1 2 3 3 4 2 6 1
RCT34 rid card 1 1 2 2 5 1 6 1
RCT26 rid card 1 2 3 2 4 1 5 1
RCT46 rid card 1 1 2 1 3 1 5 0
RCT45 rid card 1 2 2 2 3 1 6 0
RCT36 rid card 1 2 2 2 4 1 5 1
RCT35 rid card 1 1 2 3 4 1 6 1

Figure 4.3. Bottom-Up Row Enumerated Tree with Rowset Cardinality Table for respective row enumerated nodes

- *For each row in (R_2 - Y) = {2,3,5,6}, the number of ones appearing at the indices {1,2,4,8} from the preprocessed bitTable as shown in Table 3.5 are {2,1,2,1} respectively.*
- *Rowset Cardinality Table for row enumerated node 23, RCT_{23} is shown in Table 4.2d. The indices of all the ones appearing in the bitset result (01101100) at row*

enumerated node 23 are $\{2,3,5,6\}$.

- *For each row in (R_2 - Y) = $\{1,4,5,6\}$, the number of ones appearing at the indices $\{2,3,5,6\}$ from the preprocessed bitTable as shown in Table 3.5 are $\{2,2,3,2\}$ respectively.*

4.4.2 Proposed Closeness Checking

The RCT based closeness checking method is proposed to speed up the closeness checking of a rowset during the traversal of bottom-up row enumerated tree. The closeness of the rowset indicates that the itemset occurring in that rowset is also closed. The proposed efficient closeness checking method will not scan through the previously mined FCCI to check the existence and closeness of newly mined frequent colossal itemset.

Lemma 1. *A rowset $Y \subseteq R_{final}$ during the row enumeration is closed iff the cardinality of all the rows in the RCT_Y is less than the cardinality of an itemset $X \subseteq F_{final}$ mined at rowset Y.*

Proof. According to definition 8 (equations 1.1, 1.2, 1.3 and 1.4), if rowset Y is closed, then Y=$r(X)$. It is necessary to prove that Y=$r(X)$ with the help of RCT_Y. The $RCT_Y(rid)$ returns the updated cardinality value corresponding to the *rid* in RCT_Y. (For all)$\forall$ *rid* $\in$ (R_{final} - Y), if $RCT_Y(rid)$ is less than the cardinality of an itemset X mined at the rowset Y, then an itemset X has not occurred in (R_{final} - Y). This indicates that an itemset X has occurred only in Y, hence Y=$r(X)$ proved. Therefore, rowset Y is closed.

Lemma 2. *A rowset $Y \subseteq R_{final}$ during the row enumeration is not closed iff the cardinality of any one of the rows in the RCT_Y is equal to the cardinality of an itemset $X \subseteq F_{final}$ mined at a rowset Y.*

Proof. According to definition 8, if rowset Y is not closed, then $Y \neq r(X)$. It is necessary to prove that $Y \neq r(X)$ with the help of RCT_Y. The $RCT_Y(rid)$ returns the updated cardinality value corresponding to the *rid* in RCT_Y. According to the steps followed for obtaining RCT_Y at rowset Y, the updated cardinality for each *rid* in RCT_Y will never be greater than the cardinality of an itemset X mined at rowset Y. (For any)$\forall$ *rid* $\in$ (R_{final} - Y), if $RCT_Y(rid)$ is equal to the cardinality of an itemset X mined at the rowset Y, then an itemset X has occurred in (R_{final} - Y). Hence, $Y \neq r(X)$ proved. Therefore, rowset Y is not closed.

The RCT based closeness checking method is based on lemma 1 and lemma 2. If the rowset is closed then the itemset occurring in that rowset is also closed. An itemset *abd* is obtained from rowset *12* during row enumeration. The rowset *12* is closed because the cardinality of all the rows in RCT_{12} as shown in Table 4.2a is less than the cardinality of *abd*. Hence *abd* is also closed. An itemset *bd* is obtained from rowset *13* during row enumeration. The rowset *13* is not closed because the cardinality of *rid 2* and *rid 6* in RCT_{13} as shown in table 4.2b is equal to the cardinality of *bd*.

4.4.3 Proposed Pruning Strategy

The RCT based pruning strategy is proposed to efficiently cut down the row enumerated search space. The proposed pruning strategy utilizes RCT at every row enumerated node as it provides the prior information regarding the cardinality of an itemset to be mined at descendant row enumerated nodes without traversing them, unlike existing FCCI mining algorithms which does not provide any prior information regarding the same.

Table 4.3. Rowset Cardinality Table of row enumerated node *123*, RCT_{123}

rid	card
4	1
5	1
6	2

Given a rowset *Y*, if the cardinality of any *rid*s' in RCT_Y is less than the *mincard*, then the descendant row enumerated nodes with respect to those *rid*s' can be pruned as they do not contribute to the mining of colossal itemsets. For example, with *minsup* and *mincard* values set to 2, the RCT_{123} for a row enumerated node *123* as shown in Table 4.3 gives the prior information regarding the cardinality of itemsets to be mined at descendant row enumerated nodes (*1234*, *1235* and *1236*). The cardinality of *rid 4* and *rid 5* in RCT_{123} are less than *mincard*, which leads to the pruning of row enumerated nodes *1234*, *12345*, *123456*, *12346*, *1235* and *12356* as they do not contribute to the mining of colossal itemsets. The existing algorithms lack the ability to retrieve the prior information regarding the cardinality of an itemset to be mined at descendant row enumerated nodes, while the proposed RCT based pruning strategy overcomes this drawback.

4.4.4 BitSet Frequent Colossal Closed Itemset Mining (BSFCCIM) algorithm

The BSFCCIM algorithm mines the complete set of FCCI from the high dimensional dataset. Algorithm 4.3 highlights the BSFCCIM algorithm, which invokes procedure 4.3a to mine FCCI. The procedure 4.3a shows that the BSFCCIM procedure consists of RCT based efficient closeness checking method and efficient pruning strategies. The BSFCCIM algorithm mines the FCCI by performing the depth-first traversal of bottom-up row enumerated tree. The preprocessed bitTable, *minsup*, and *mincard* are provided as input to the BSFCCIM algorithm. The FCCI, set of frequent colossal closed itemsets is initialized to null. R'_{final} is the set of rows to be enumerated. The *rcomb* is the initial row enumerated node considered during the depth-first traversal of bottom-up row enumerated tree. The BSFCCIM procedure is invoked to mine the frequent colossal closed itemsets from preprocessed bitTable.

- **Pruning Strategy 1:** If the row enumerated node *rcomb* or its descendant row enumerated nodes does not reach till *minsup* then, the *rcomb* and its descendants if existing, is pruned to cut down the row enumerated search space. For example,

Algorithm 4.3. BSFCCIM algorithm

```
Input: preprocessed bitTable, minsup, mincard.
Output: Complete set of Frequent Colossal Closed Itemsets,FCCI
    Initialisation: FCCI= ∅
                    $R'_{final}$ = set of rows to be enumerated
                    rcomb = initial node in row enumeration
                    All bit's in bitset_result are initialized to 1
1:    BSFCCIM(rcomb,bitset_result,$R'_{final}$,FCCI)
```

Procedure 4.3a. BSFCCIM(*rcomb*,bitset_result,R'_{final},FCCI)

```
1:  Pruning 1: if node rcomb or its descendants does not reach till minsup
2:          return
3:  calculate bitset_result at the node rcomb
4:  Pruning 2: $\forall$ rid $\in R'_{final}$, if $RCT_{rcomb}(rid) < mincard$
5:          delete rid from $R'_{final}$
6:  Optimization: if $|rcomb| < minsup$
7:            discard closeness checking
8:        else
9:            if Closeness_Checking(itemset,rcomb) == True
10:               Add itemset to FCCI
11: for each row combination(rcomb) in $R'_{final}$
12:   BSFCCIM(rcomb,bitset_result,$R'_{final}$,FCCI)
```

the row enumerated node *6* shown in Figure 4.2 will be pruned during the mining process if the *minsup* value is set to 2, as it will not reach to *minsup* level. The row enumerated nodes *46*, *5*, *56*, *6* shown in Figure 4.2 are pruned during the mining process if the *minsup* value is set to 3.

- The bitset_result is calculated at row enumerated node *rcomb*. For example, the bitset_result at row enumerated node *12* shown in Figure 4.2 is 11100000 (*abd*).

- **Pruning Strategy 2:** Pruning strategy 2 in BSFCCIM algorithm highlight the proposed RCT based pruning strategy. The RCT based pruning strategy provides an added computational boost to the proposed BSFCCIM algorithm as it provides the prior information regarding the cardinality of the itemsets to be mined at descendant row enumerated nodes, whereas the existing FCCI mining algorithms does not provide the prior information regarding the same.

- **Optimization:** If the number of *rids*' in *rcomb* is less than *minsup* then the closeness checking of the rowset *rcomb* is not required. The optimization in BSFCCIM helps to skip closeness checking for (*minsup*-1) number of levels. For example, the closeness checking of all the *1*-rowsets and *2*-rowsets is not required when the *minsup* value is set to 3.

- **Closeness Checking:** The procedure 4.3b and step 9 in procedure 4.3a highlight the proposed RCT based closeness checking method of a rowset. The proposed RCT based closeness checking method is based on lemma 1 and lemma 2. If the rowset satisfies the closeness checking then the itemset mined from that rowset is added to FCCI. The algorithm continues with depth-first traversal of bottom-up row enumerated tree.

Procedure 4.3b. Closeness_Checking(itemset,*rcomb*)

```
1: Let $R_{final}$ be set of rows in preprocessed table
2: (for any)$\forall$ $rid \in (R_{final}$ - $rcomb)$, if $RCT_{rcomb}(rid)$==card($itemset$)
3:   flag_closed = false
4:   break
5: if flag_closed == false
6:   return False
7: else
8:   return True
```

4.4.5 Complexity Analysis

For the high dimensional dataset, let R_{final} be the number of rows and F_{final} be the number of features after applying the proposed preprocessing technique. The space complexity of the bitTable is $\mathcal{O}(R_{final}F_{final})$. The rowset closeness checking method and pruning strategy of the proposed BSFCCIM algorithm will take advantage of the RCT at the respective row enumerated node. During the row enumeration approach, the RCT at any particular row enumerated node Y will be in the memory until the completion of rowset closeness checking method and pruning strategy. Hence there will be only one RCT in the memory during the row enumeration approach and requires $\mathcal{O}(R_{final} - |Y|)$ to be in the memory. The space complexity during the row enumeration approach is $\mathcal{O}(R_{final}F_{final} + (R_{final} - |Y|))$.

The proposed BSFCCIM algorithm with row enumeration approach traverse all the row enumerated nodes in the worst case. The total number of row enumerated nodes that need to be traversed in the worst case is u, $u=\sum_{l=1}^{R_{final}} {}^{R_{final}}C_l$. The time required for the RCT based rowset closeness checking method and pruning strategy is $\mathcal{O}(R_{final} - |Y|)$. The time complexity is $\mathcal{O}(u(R_{final} - |Y|))$ during the row enumeration approach. Let the total number of row enumerated nodes that need to be traversed in average case be, c, such that $c=\sum_{l=1}^{k} {}^{k}C_l$, where k is the level in the bottom-up row enumerated tree up to which all the mined itemsets are colossal; all the nodes that are present in the levels higher than k will not be traversed as these levels do not contribute for the mining of colossal itemsets; $k << R_{final}$ and $c << u$. The time complexity is $\mathcal{O}(c(R_{final} - |Y|))$ during the row enumeration approach in the average case.

4.5 Results and Discussion

This section emphasizes on the better accuracy achieved by the proposed frequent colossal itemset mining algorithm compared to the existing algorithm in terms of a number of frequent colossal itemsets mined from the high dimensional dataset. This section also highlights the efficiency of the proposed frequent colossal closed itemset mining algorithm and BSFCCIM algorithm. The experiments were conducted on a computer with a specification of 3.4GHz core i7-3770 CPU, 8GB RAM, and 1TB hard disk.

4.5.1 Results of Frequent Colossal Itemset Mining from the High Dimensional Dataset

The proposed frequent colossal itemset mining algorithm has been applied on lung cancer test and Acute Lymphoblastic Leukemia-Acute Myeloid Leukemia (ALL-AML) datasets. The details of the high dimensional datasets have been explained in section 3.3 of chapter 3. The proposed frequent colossal itemset mining algorithm has been compared with existing BVBUC algorithm. Pattern Fusion algorithm is based on feature enumeration approach, and it is best suited to mine the frequent colossal itemsets from transactional datasets. Feature enumeration based Pattern Fusion algorithm face an uphill task in mining frequent colossal itemsets from the high dimensional dataset.

The BVBUC algorithm is designed to mine the frequent colossal itemsets from the high dimensional dataset. Hence it is chosen as a representative for the performance evaluation. The BVBUC algorithm mine the itemsets from the nodes belonging to the minimum support threshold level of a row enumerated tree and prune their descendant row enumerate nodes. The BVBUC algorithm will not be able to mine complete set of frequent colossal itemsets from the high dimensional dataset. The proposed frequent colossal itemset mining algorithm and BVBUC algorithm has been implemented in C++. Equation 4.1 shows the percentage of frequent colossal itemsets mined by the BVBUC algorithm. Equation 4.2 shows the percentage of frequent colossal itemsets mined by the proposed algorithm. In both the equations FP-growth (Han *et al.* (2000)) is used as a base method to find the number of frequent colossal itemsets.

$$\%\ of\ frequent\ colossal\ itemsets\ mined\ by\ BVBUC = (\frac{E_{BVBUC}}{T_{FP}}) \times 100 \tag{4.1}$$

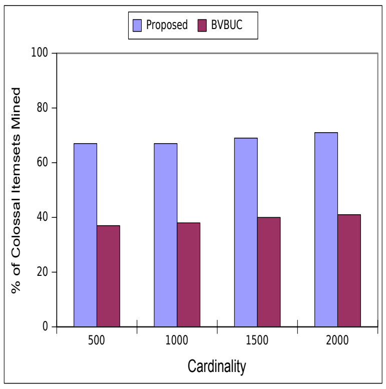
$$\%\ of\ frequent\ colossal\ itemsets\ mined\ by\ Proposed = (\frac{P_{Proposed}}{T_{FP}}) \times 100 \tag{4.2}$$

where:

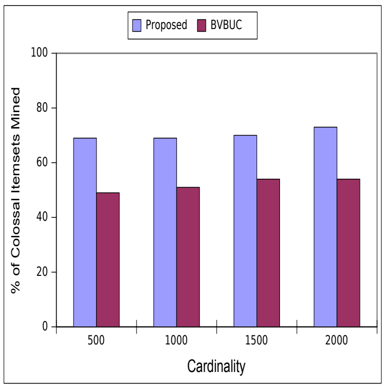
T$_{FP}$	=	number of frequent colossal itemsets mined by FP-growth for given *minsup* and *mincard*
E$_{BVBUC}$	=	number of frequent colossal itemsets mined by BVBUC for given *minsup* and *mincard*
P$_{Proposed}$	=	number of frequent colossal itemsets mined by proposed algorithm for given *minsup* and *mincard*

Figure 4.4 and Figure 4.5 shows the accuracy of proposed frequent colossal itemset mining algorithm and BVBUC algorithm. Figure 4.4a and Figure 4.4b shows the accuracy in terms of a number of frequent colossal itemsets mined from lung cancer test dataset when *minsup* value is set to 5 and 10 respectively. Figure 4.5a and Figure 4.5b shows the accuracy in terms of a number of frequent colossal itemsets mined from ALL-AML dataset when *minsup* value is set to 5 and 10 respectively. The *x*-axis in the Figure 4.4 and Figure 4.5 indicates the different values of *mincard*. The *y*-axis in the the Figure 4.4 and Figure 4.5 indicates the percentage of frequent colossal itemsets mined by the respective algorithm.

The proposed algorithm achieves better accuracy than BVBUC algorithm in terms of mining number of frequent colossal itemsets. The proposed algorithm achieved an accuracy of 67% and 68% for the lung cancer test dataset, as shown in Figure 4.4a, when the (*minsup*, *mincard*) values are set to (5, 500) and (5, 1000) respectively. The proposed algorithm outperforms the BVBUC algorithm by 30% and 29% for the lung

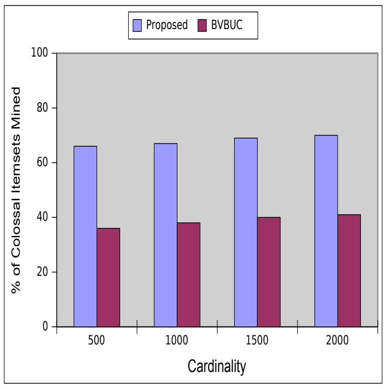


(a) *minsup*=5

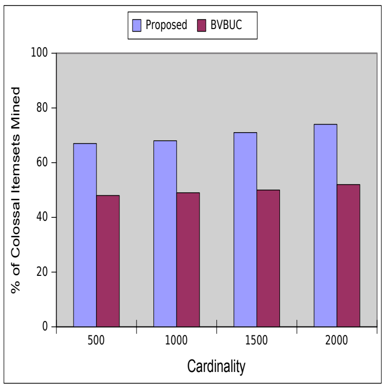


(b) *minsup*=10

Figure 4.4. Accuracy of Proposed and BVBUC algorithm for Lung Cancer Test Dataset When the *minsup* is set to 5 and 10



(a) *minsup*=5



(b) *minsup*=10

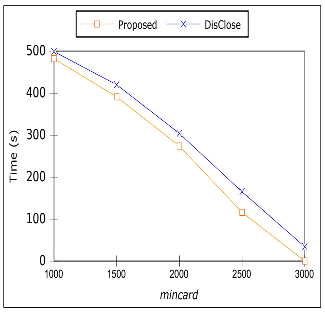
Figure 4.5. Accuracy of Proposed and BVBUC algorithm for ALL-AML Dataset When the *minsup* is set to 5 and 10

cancer test dataset, when the (*minsup*, *mincard*) values are set to (5, 500) and (5, 1000) respectively. Similar results have been observed for different values of *mincard*. The proposed algorithm achieved an accuracy of 69% and 70% for the lung cancer test dataset, as shown in Figure 4.4b, when the (*minsup*, *mincard*) values are set to (10, 500) and (10, 1000) respectively. The proposed algorithm outperforms the BVBUC algorithm by 20% and 19% for the lung cancer test dataset, when the (*minsup*, *mincard*) values are set to (10, 500) and (10, 1000) respectively. Similar results have been observed for different values of *mincard*.

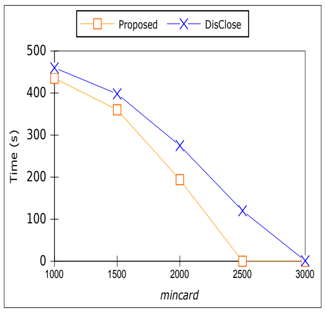
The proposed algorithm achieved an accuracy of 66% and 67% for the ALL-AML dataset as shown in Figure 4.5a, when the (*minsup*, *mincard*) values are set to (5, 500) and (5, 1000) respectively. The proposed algorithm outperforms the BVBUC algorithm by 28% and 28% for the ALL-AML dataset, when the (*minsup*, *mincard*) values are set to (5, 500) and (5, 1000) respectively. Similar results have been observed for different values of *mincard*. The proposed algorithm achieved an accuracy of 68% and 70% for the ALL-AML dataset as shown in Figure 4.5b, when the (*minsup*, *mincard*) values are set to (10, 500) and (10, 1000) respectively. The proposed algorithm outperforms the BVBUC algorithm by 20% and 21% for the ALL-AML dataset, when the (*minsup*, *mincard*) values are set to (10, 500) and (10, 1000) respectively. Similar results have been observed for different values of *mincard*. The proposed frequent colossal itemset mining algorithm achieves better accuracy as it mines the itemsets from the nodes belonging to the *minsup* level of row enumerated tree and from its descendant row enumerated nodes which contribute to the mining of frequent colossal itemsets.

4.5.2 Results of Frequent Colossal Closed Itemset Mining using Prune Table

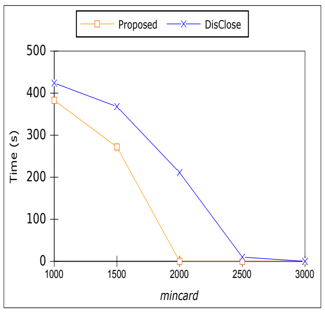
The proposed frequent colossal closed itemset mining algorithm has been applied on Diffuse Large B-Cell Lymphoma (DLBCL) and lung cancer test datasets. The details of the high dimensional datasets have been explained in section 3.3 of chapter 3. The proposed frequent colossal closed itemset mining algorithm has been compared with the DisClose algorithm. The DisClose algorithm outperforms the other existing FCCI mining algorithms. Hence it is chosen as representative for the experimental evaluation. The Pattern Fusion and BVBUC algorithm fail to mine the complete set of FCCI from the high dimensional dataset. Hence the Pattern Fusion and BVBUC algorithm cannot be considered for the experimental runtime evaluation. The proposed frequent colossal closed itemset mining algorithm and DisClose algorithm have been implemented in C++.



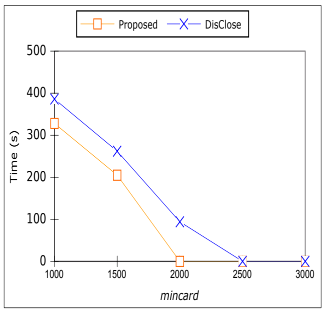
(a) minsup=5



(b) minsup=10



(c) minsup=15

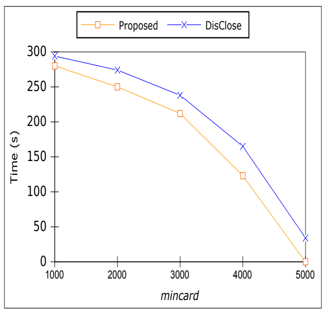


(d) minsup=20

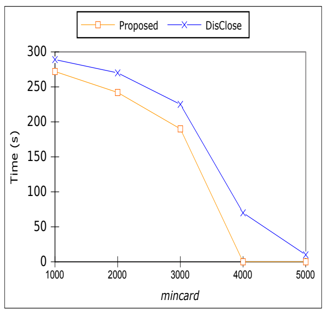
Figure 4.6. Runtime of Proposed Frequent Colossal Closed Itemset Mining Algorithm and DisClose Algorithm for DLBCL Dataset

Figure 4.6 and Figure 4.7 shows the runtime of the proposed frequent colossal closed itemset mining algorithm and DisClose algorithm at different *minsup* and different *mincard* for DLBCL and lung cancer test dataset, respectively. The *x*-axis in Figure 4.6 and Figure 4.7 indicates the different values of *mincard*. The *y*-axis in Figure 4.6 and Figure 4.7 indicates the runtime. It has been observed from the experimental results that the runtime of proposed frequent colossal closed itemset mining algorithm reduces as the *minsup* and *mincard* increases. The experimental results as shown in Figure 4.6 and Figure 4.7 indicate that the proposed frequent colossal closed itemset mining algorithm is not obligatory to gauge the final result when the number of significant rows and significant features reach zero after applying the proposed EIP technique for a given dataset, *minsup* and *mincard*.

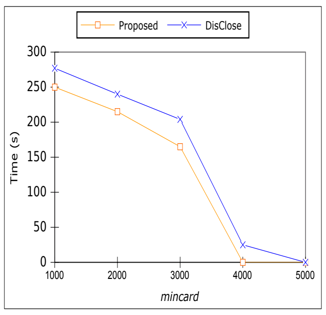
The experimental results highlight that the proposed frequent colossal closed itemset mining algorithm outperforms Disclose algorithm in terms of runtime. The proposed frequent colossal closed itemset mining algorithm outperforms the DisClose algorithm by (17, 25, 41, 58) seconds for the DLBCL dataset, as shown in Figure 4.6, when the (*minsup*, *mincard*) values are set to (5, 1000), (10, 1000), (15, 1000) and (20, 1000) respectively. Similar results have been observed for different values of (*minsup*, *mincard*). The proposed frequent colossal closed itemset mining algorithm outperforms



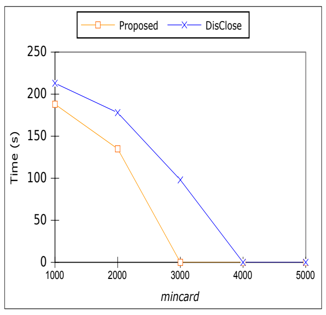
(a) minsup=5



(b) minsup=10



(c) minsup=15



(d) minsup=20

Figure 4.7. Runtime of Proposed Frequent Colossal Closed Itemset Mining Algorithm and DisClose Algorithm for Lung Cancer Test Dataset

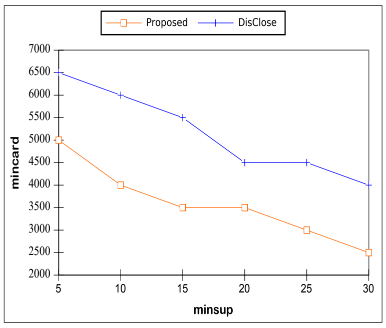
the DisClose algorithm by (14, 17, 27, 25) seconds for the lung cancer test dataset, as shown in Figure 4.7, when the (*minsup*, *mincard*) values are set to (5, 1000), (10, 1000), (15, 1000) and (20, 1000) respectively. Similar results have been observed for different values of (*minsup*, *mincard*). The PT based pruning strategy enriched in proposed frequent colossal closed itemsets mining algorithm is efficient in cutting down the row enumerated search space. Figure 4.6 and Figure 4.7 highlights that the runtime decreases as the *minsup* and *mincard* increases.

4.5.3 Results of BitSet Frequent Colossal Closed Itemset Mining (BSFCCIM) algorithm

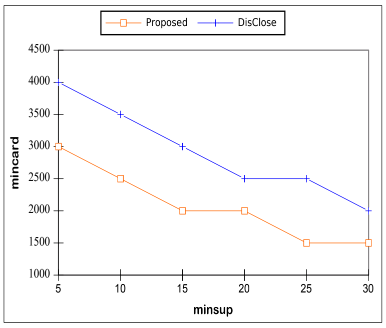
The proposed BSFCCIM algorithm has been applied on Mixed Lineage Leukemia (MLL), Central Nervous System embryonal tumor and Diffuse Large B-Cell Lymphoma (DLBCL) including Follicular Lymphoma datasets. The details of the high dimensional datasets have been explained in section 3.3 of chapter 3. The proposed BSFCCIM algorithm has been compared with the DisClose algorithm. The DisClose algorithm outperforms the other existing FCCI mining algorithms. Hence it is chosen as representative for the experimental evaluation. The Pattern Fusion and BVBUC algorithm fail to mine the complete set of FCCI from the high dimensional dataset. Hence the Pattern Fusion and BVBUC algorithm cannot be considered for the experimental runtime evaluation. The proposed BSFCCIM algorithm and DisClose algorithm have been implemented in C++.

The Figure 4.8a, Figure 4.8b and Figure 4.8c highlights the *minsup* and *mincard* at which the number of significant rows has been zero after applying proposed EIP technique and preprocessing technique of DisClose algorithm for MLL, central nervous system embryonal tumor and DLBCL (including Follicular Lymphoma) datasets respectively. The *x*-axis in the Figure 4.8 indicates the different values of *minsup*. The *y*-axis in the Figure 4.8 indicate the different values of *mincard*. The Figure 4.8a, Figure 4.8b and Figure 4.8c also shows that, after preprocessing the number of significant rows reach zero for lesser *mincard* as the *minsup* increases.

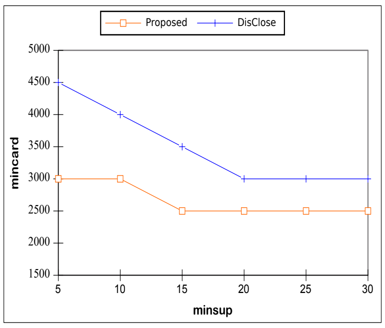
After applying the proposed EIP technique, the number of significant rows and significant features in the MLL dataset is zero when the *minsup* value reach 5, and *mincard* value reach 5000, as shown in Figure 4.8a. This indicates that BSFCCIM algorithm is



(a) Different *minsup* and *mincard* at which number of significant rows is zero after applying the proposed EIP technique and preprocessing technique of DisClose algorithm for MLL dataset



(b) Different *minsup* and *mincard* at which number of significant rows is zero after applying the proposed EIP technique and preprocessing technique of DisClose algorithm for Central Nervous System embryonal tumor Dataset

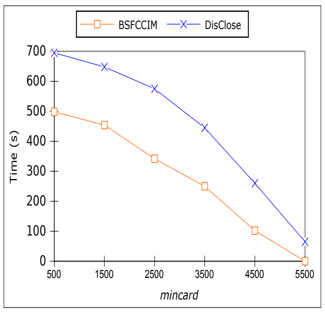


(c) Different *minsup* and *mincard* at which number of significant rows is zero after applying the proposed EIP technique and preprocessing technique of DisClose algorithm for DLBCL (Including Follicular Lymphoma) Dataset

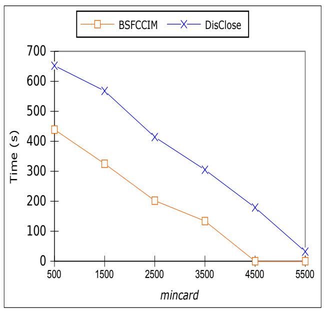
Figure 4.8. Different *minsup* and *mincard* at which number of significant rows is zero after applying the proposed EIP technique and preprocessing technique of DisClose algorithm for different datasets

not obligatory to gauge the final result for MLL dataset when the *minsup* value reach 5, and *mincard* value reach 5000, whereas DisClose algorithm has to enumerate through huge row space to gauge the final result at the same *minsup* and *mincard*. After applying the proposed EIP technique, the number of significant rows and significant features in the Central Nervous System embryonal tumor dataset is zero when the *minsup* value reach 5, and *mincard* value reach 3000, as shown in Figure 4.8b. This indicates that proposed BSFCCIM algorithm is not obligatory to gauge the final result for Central Nervous System embryonal tumor dataset when the *minsup* value reach 5, and *mincard* value reach 3000. But, the DisClose algorithm is not obligatory to gauge the final result for Central Nervous System embryonal tumor dataset when the *minsup* value reach 5, and *mincard* value reach 4000. Similarly, the proposed BSFCCIM algorithm is not obligatory to gauge the final result for DLBCL (including Follicular Lymphoma) dataset when the *minsup* value reach 5, and *mincard* value reach 3000, as shown in Figure 4.8c.

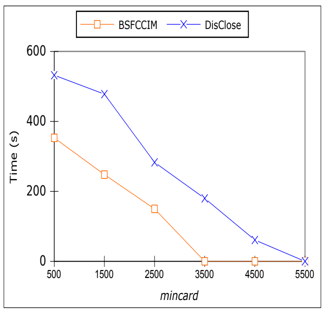
Figure 4.9, Figure 4.10 and Figure 4.11 shows the runtime of the proposed BSFCCIM algorithm and DisClose algorithm at different *minsup* and different *mincard* for MLL, central nervous system embryonal tumor and DLBCL (including Follicular Lymphoma) datasets respectively. The *x*-axis in the Figure 4.9, Figure 4.10 and Figure



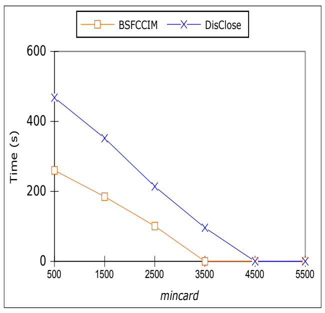
(a) minsup=5



(b) minsup=10

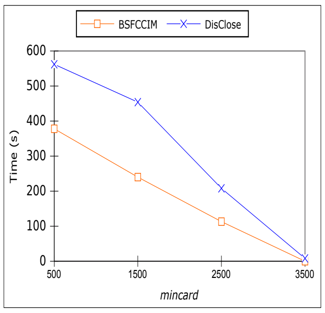


(c) minsup=15

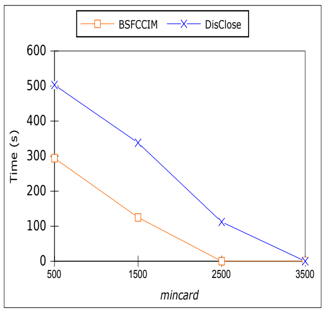


(d) minsup=20

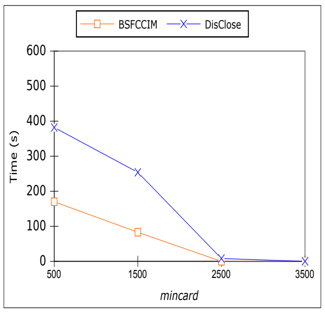
Figure 4.9. Runtime of BSFCCIM Algorithm and DisClose Algorithm for MLL Dataset



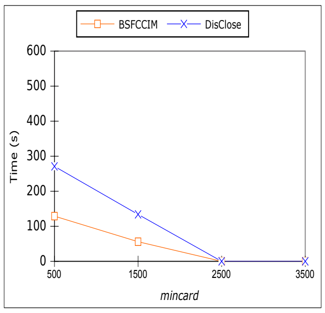
(a) minsup=5



(b) minsup=10

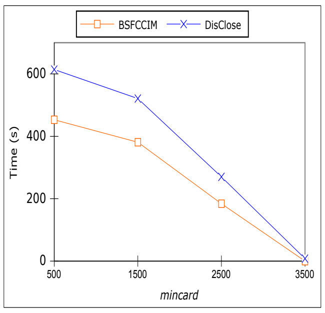


(c) minsup=15

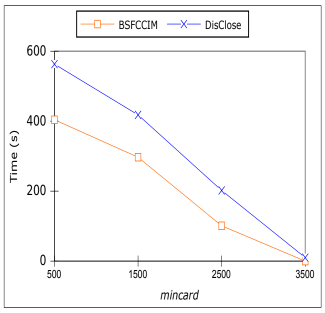


(d) minsup=20

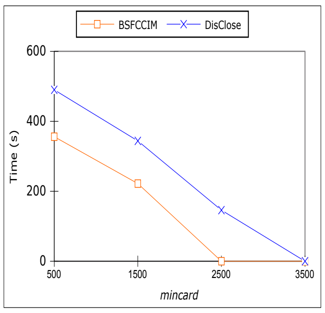
Figure 4.10. Runtime of BSFCCIM Algorithm and DisClose Algorithm for Central Nervous System embryonal tumor Dataset



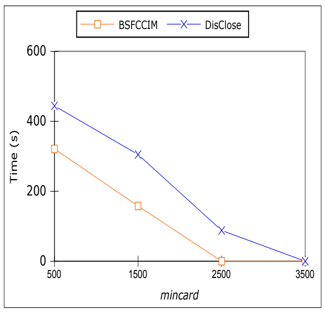
(a) minsup=5



(b) minsup=10



(c) minsup=15



(d) minsup=20

Figure 4.11. Runtime of BSFCCIM Algorithm and DisClose Algorithm for DLBCL (Including Follicular Lymphoma) Dataset

4.11 indicates different values of *mincard*. The *y*-axis in the Figure 4.9, Figure 4.10 and Figure 4.11 indicates the runtime. It has been observed from the experimental results that the runtime of the proposed BSFCCIM algorithm reduces as the *mincard* and *minsup* increases. It has been observed from the experimental results that the proposed BSFCCIM algorithm outperforms Disclose algorithm in terms of runtime for all the experimental high dimensional datasets.

The proposed BSFCCIM algorithm outperforms the DisClose algorithm by (197, 213, 179, 208) seconds for the MLL dataset, as shown in Figure 4.9, when the (*minsup*, *mincard*) values are set to (5, 500), (10, 500), (15, 500) and (20, 500) respectively. Similar results have been observed for different values of (*minsup*, *mincard*). The proposed BSFCCIM algorithm outperforms the DisClose algorithm by (184, 209, 212, 143) seconds for the central nervous system embryonal tumor dataset, as shown in Figure 4.10, when the (*minsup*, *mincard*) values are set to (5, 500), (10, 500), (15, 500) and (20, 500) respectively. Similar results have been observed for other experimental high dimensional datasets for different values of (*minsup*, *mincard*). The proposed RCT based closeness checking method and pruning strategy enclosed in the proposed BSFCCIM algorithm helps it to efficiently mine the FCCI from the high dimensional dataset compared to the DisClose algorithm. The proposed RCT based closeness checking method helps to efficiently check the closeness of the rowset and RCT based pruning strategy helps to efficiently prune the row enumerated search space.

4.6 Summary

In this chapter, frequent colossal itemset mining algorithm has been proposed to achieve better accuracy than existing algorithms in terms of mining number of frequent colossal itemsets from the high dimensional dataset. The proposed frequent colossal itemset mining algorithm outperforms the existing algorithms by achieving better accuracy in mining a greater number of frequent colossal itemsets from the high dimensional dataset. The proposed algorithm outperforms the BVBUC algorithm by 30% and 29% for the lung cancer test dataset, when the (*minsup*, *mincard*) values are set to (5, 500) and (5, 1000) respectively.

The BitSet Frequent Colossal Closed Itemset Mining (BSFCCIM) algorithm enclosed with RCT based closeness checking method and pruning strategy has been proposed to mine the FCCI from the high dimensional dataset. The experiments have been conducted on various high dimensional datasets. It is observed from the experiment results that the proposed BSFCCIM algorithm outperforms the existing DisClose algorithm in terms of runtime. The proposed BSFCCIM algorithm outperforms the DisClose algorithm by (197, 213, 179, 208) seconds for the MLL dataset, when the (*minsup*, *mincard*) values are set to (5, 500), (10, 500), (15, 500) and (20, 500) respectively. Similar results have been observed for other experimental high dimensional datasets.

In the next chapter, the dynamic switching frequent colossal closed itemset mining algorithm has been discussed.

Chapter 5

Dynamic Switching Algorithm for Mining Frequent Colossal Closed Itemsets

The existing algorithms are computationally expensive in mining FCCI from datasets consisting of a large number of rows and a large number of features, as they are either pure row or feature enumeration based algorithms. The existing frequent colossal closed itemset mining algorithms are inefficient in handling the changing characteristics of the data subset during the mining process. To overcome the drawbacks, an efficient Dynamic Switching Frequent Colossal Closed Itemset Mining (DSFCCIM) algorithm has been proposed. The proposed DSFCCIM algorithm efficiently switches between bottom-up row enumerated approach and bottom-up feature enumerated approach based on data characteristics during the mining process.

5.1 Search Strategies

A pure row or feature enumerated approach is inefficient as the characteristics of the data subset changes from one subset to another. A combination of search strategies is required to efficiently handle the different characteristics possessed by different data subsets during the mining process. An enumerated tree can either be a row enumerated tree or a feature enumerated tree. The proposed DSFCCIM algorithm efficiently switches between bottom-up row enumerated approach and bottom-up feature enumerated approach based on data characteristics during the mining process. The reason for the selection of a bottom-up search strategy to traverse the row enumerated tree is explained in section 4.1 of chapter 4. This section presents the motivation for the selection of a bottom-up search strategy to traverse the feature enumerated tree.

5.1.1 Top-Down Traversal of Feature Enumerated Tree

The feature enumerated tree can be traversed by either bottom-up or top-down search strategy. The top-down traversing of the feature enumerated space implies that the search starts from larger itemset and builds smaller itemsets during the process. Figure 5.1 shows the top-down feature enumerated tree for the preprocessed bitTable 3.6, with each feature enumerated node representing the itemset and the corresponding bitset result is indicated under the corresponding feature enumerated node. The bitset at each feature enumerated node helps in determining the support of an itemset and is obtained by performing bitwise AND operations of bitset that corresponds to the features in the itemset. For example, the bitset at feature enumerated node *abd* (11000), as shown in Figure 5.1 is obtained by performing the bitwise AND operations of bitset that corresponds to the features *a* (11011), *b* (11110) and *d* (11101). The top-down traversal of the feature enumerated tree is inefficient with bitset approach as the bitset result at a parent feature enumerated node cannot be utilized to generate bitset result at the child feature enumerated node. The number of bitwise AND operations required to obtain the bitset result at the feature enumerated node is (|itemset|- 1). For example, in Figure 5.1, the bitset result at feature enumerated node *abdg* requires 3 bitwise AND operations. The top-down approach fails to take advantage of the anti-monotone property of minimum support threshold. These disadvantages of top-down traversal of feature enumerated tree make it as an inefficient search strategy for mining FCCI from dataset

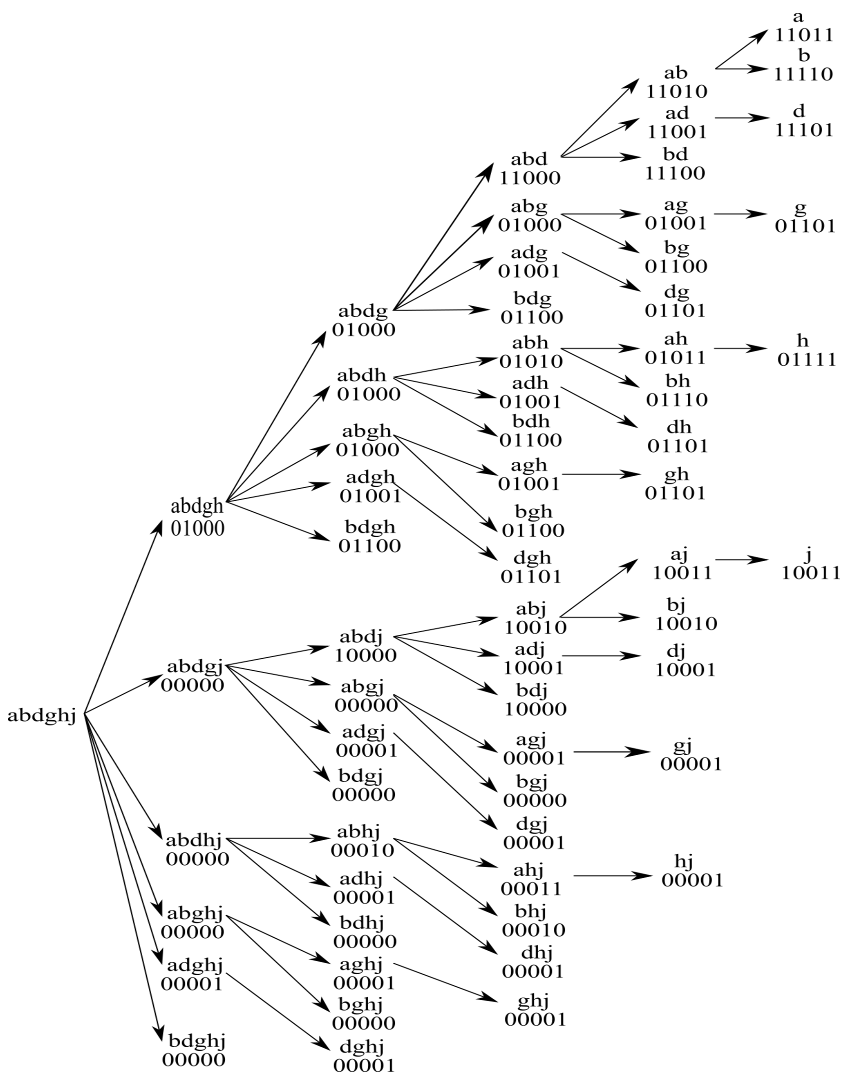


Figure 5.1. Top-Down Traversal of Feature Enumerated Tree

consisting of a large number of rows and a large number of features.

5.1.2 Bottom-Up Traversal of Feature Enumerated Tree

The bottom-up traversing of the feature enumerated space implies that the search starts from the smaller itemset and builds larger itemsets during the process. Figure 5.2 shows the bottom-up feature enumerated tree for the bitTable 3.6, with each feature enumerated node representing the itemset and the corresponding bitset result is indicated under the corresponding feature enumerated node. The bottom-up approach has a benefit of utilizing the bitset result of the parent feature enumerated node to obtain the bitset result at the child feature enumerated node, thus exponentially reducing the number of bitwise AND operations to be performed. Only one bitwise AND operation is required to obtain the bitset result at each feature enumerated node as compared to (|itemset|-1)

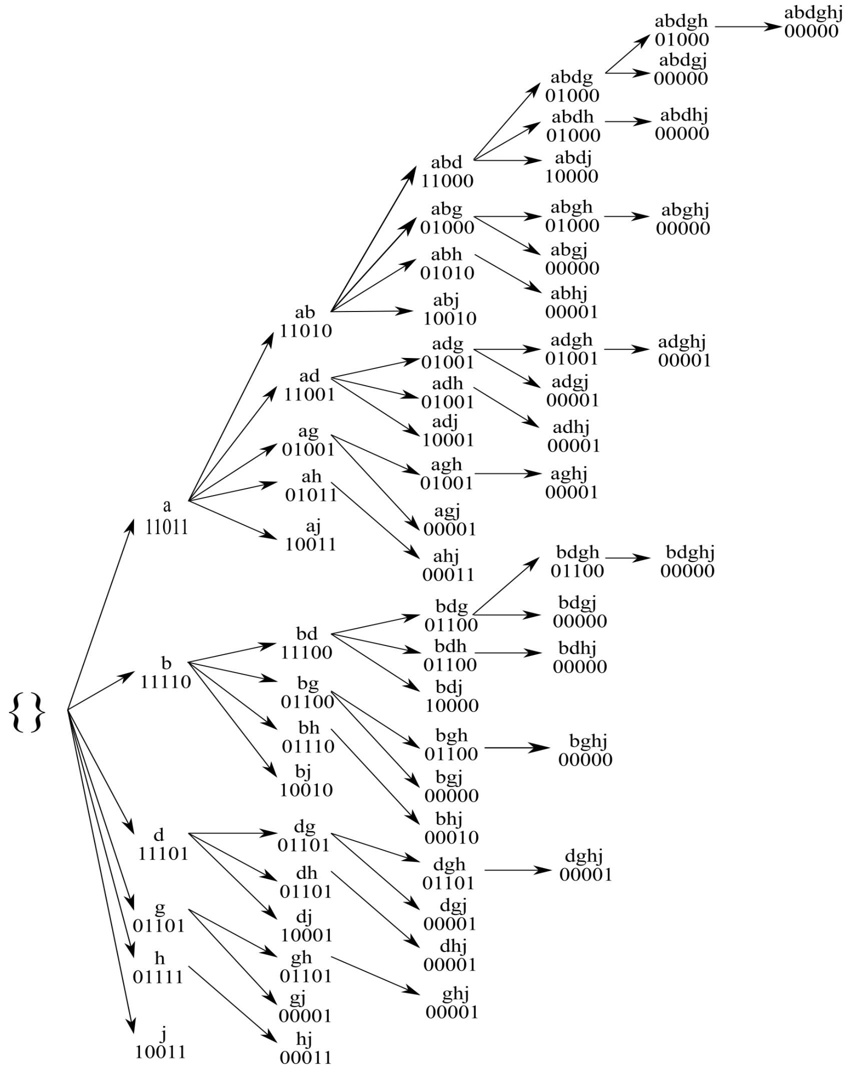


Figure 5.2. Bottom-Up Traversal of Feature Enumerated Tree

in the top-down traversal of feature enumerated tree. For example, in Figure 5.2, the bitset result at feature enumerated node *bdgh* requires one bitwise AND operation. The bitset result of feature enuemrated node *bdg* and *h* are utilized to generate the bitset result of feature enumerated node *bdgh*.

The anti-monotone property of minimum support threshold is utilized by the bottom-up search strategy to cut down the feature enumerated search space. It means that, if an bitset result at a feature enumerated node represented by *l*-itemset is not frequent, then the bitset result at a child feature enumerated node represented by (*l*+1)-itemset is also not frequent. For example, with *minsup* value set to 2 and *mincard* value set to 3, the descendants of feature enumerated node *abdg* can be pruned as the bitset result in descendant feature enumerated nodes are not frequent. The top-down approach fails to utilize the pruning power of *minsup*. The bottom-up approach will not be able to efficiently utilize the pruning power of *mincard* threshold. This disadvantage will not impede the efficiency of the bitset approach, as huge number of bitwise AND operations are required in the top-down approach compared to a bottom-up approach. Both bottom-up and top-down traversal of the feature enumerated tree have advantage and disadvantage. Depending on the advantages and disadvantages, the proposed algorithm traverses the feature enumerated tree in a bottom-up strategy using bitset approach. The DSFCCIM algorithm dynamically switches between the bottom-up row enumerated tree and bottom-up feature enumerated tree based on the data characteristics during the mining process.

5.2 Proposed Dynamic Switching Method for Mining Frequent Colossal Closed Itemsets

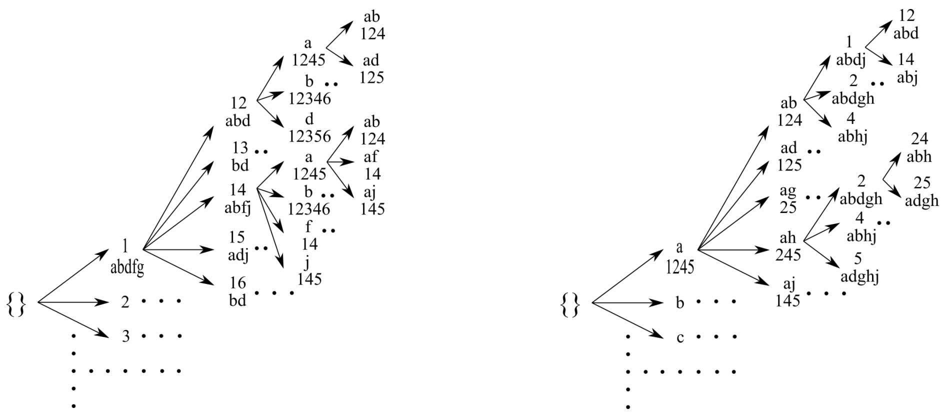
The existing frequent colossal closed itemset mining algorithms are inefficient in handling the changing characteristics of the data subset during the mining process. The closeness checking methods and pruning strategies enclosed by the existing frequent colossal closed itemset mining algorithms are inefficient. Moreover, these algorithms are either pure row enumeration or pure feature enumeration based algorithms, which are inefficient in mining FCCI from datasets consisting of a large number of rows and a large number of features. Hence there is a need to develop an algorithm with a combination of different enumeration methods to handle the changing characteristics of a data subset efficiently.

To surmount the drawbacks, an efficient Dynamic Switching Frequent Colossal

Closed Itemset Mining (DSFCCIM) algorithm integrating the following proposed techniques has been designed.

- An Effective Improved Preprocessing (EIP) technique. The details and the effectiveness of the proposed EIP technique have been described in Chapter 3.
- An efficient Rowset Cardinality Table (RCT) based closeness checking method to check the closeness of a rowset during row enumeration method. The RCT based closeness checking method has been explained in subsection 4.4.2 of chapter 4.
- An efficient RCT based pruning strategy to cut down the row enumerated mining search space by efficient utilization of the minimum cardinality threshold. The RCT based pruning strategy has been explained with examples in the subsection 4.4.3 of chapter 4.
- An efficient switching condition that dynamically switches between the bottom-up row enumeration method and bottom-up feature enumeration method to handle the changing characteristics of the data subset during the mining process.
- An efficient Itemset Support Table (IST) based closeness checking method was proposed to check the closeness of an itemset during feature enumeration method.
- An efficient IST based pruning strategy has been proposed to cut down the feature enumerated mining search space by efficient utilization of the minimum support threshold. The IST at the feature enumerated node provides prior information regarding the support of the itemsets to be mined at descendant feature enumerated nodes without traversing them, unlike existing FCCI mining algorithms which do not provide any prior information regarding the same.

The proposed DSFCCIM algorithm dynamically switches between the bottom-up row enumeration method and bottom-up feature enumeration method to handle the changing characteristics of a data subset during the mining process. Figure 5.3a shows an example of dynamic switching from bottom-up row enumerated tree to bottom-up feature enumerated tree. Figure 5.3b shows an example of dynamic switching from bottom-up feature enumerated tree to bottom-up row enumerated tree. The Itemset Support Table (IST), IST based closeness checking method, IST based pruning strategy and the proposed DSFCCIM algorithm have been explained in the following subsections.



(a) Dynamic Switching from bottom-up row enumerated tree to bottom-up feature enumerated tree
(b) Dynamic Switching from bottom-up feature enumerated tree to bottom-up row enumerated tree

Figure 5.3. Combination of bottom-up row and feature enumerated tree

5.2.1 Itemset Support Table

The proposed closeness checking method takes advantage of the Itemset Support Table (IST) in the feature enumerated tree to check the closeness of an itemset. The IST helps in closeness checking of an itemset without the need to scan through the previously mined FCCI itemsets. The proposed pruning strategy utilizes the IST to efficiently cut down the feature enumerated search space. The IST provides prior information regarding the support of the itemsets to be mined at the descendant feature enumerated nodes without traversing them. The IST for every feature enumerated node is shown in Figure 5.4. Each feature enumerated node refers their respective proposed Itemset Support Table as shown in Figure 5.4 for closeness checking of an itemset and pruning the descendant feature enumerated nodes which do not contribute for the mining of frequent itemsets.

Definition 11 (Itemset Support Table)**.** *Given a itemset X=$\{f_{i1}, f_{i2}, ..., f_{ik}\}$ representing a feature enumerated node in a lexicographical order, the Itemset Support Table (IST_X) at node X contains the updated support for each gene feature in (F_{final} - X) depending on the support of the bitset result obtained at feature enumerated node X.*

The Itemset Support Table (IST_X) at feature enumerated node *X* is obtained by the following steps.

1. Obtain the indices of all the ones appearing in the bitset result obtained at feature enumerated node *X*.
2. For each feature in (F_{final} - *X*), calculate the number of ones from the preprocessed bitTable appearing at the indices obtained from the step 1.

Example 13. *Tables 5.1a, 5.1b, 5.1c and 5.1d show the Itemset Support Table at feature enumerated nodes ab, bd, dg and ah respectively in Figure 5.4.*

Table 5.1. Itemset Support Table of feature enumerated nodes *ab*, *bd*, *dg* and *ah*

(a) IST_{ab}

feature	sup
d	2
g	1
h	2
j	2

(b) IST_{bd}

feature	sup
a	2
g	2
h	2
j	1

(c) IST_{dg}

feature	sup
a	2
b	2
h	3
j	1

(d) IST_{ah}

feature	sup
b	2
d	2
g	2
j	2

Example 14. *Obtaining Itemset Support Table for feature enumerated nodes ab (IST_{ab}) and dg (IST_{dg}) in Figure. 5.4.*

- *Itemset Support Table for feature enumerated node ab, IST_{ab} is shown in Table 5.1a. The indices of all the ones appearing in the bitset result (11010) at feature enumerated node ab are {1,2,4}.*

- *For each feature in {d,g,h,j}, the number of ones appearing at the indices {1,2,4} from the preprocessed bitTable as shown in Table 3.6 are {2,1,2,2} respectively.*

- *Itemset Support Table for feature enumerated node dg, IST_{dg} is shown in Table 5.1c, the indices of all the ones appearing in the bitset result (01101) at node feature enumerated dg are {2,3,5}.*

- *For each row in {a,b,h,j}, the number of ones appearing at the indices {2,3,5} from the preprocessed bitTable as shown in Table 3.6 are {2,2,3,1} respectively.*

5.2.2 Proposed IST based Closeness Checking

The IST based closeness checking method is proposed to speed up the closeness checking of an itemset during the traversal of a feature enumerated tree. The proposed efficient closeness checking method will not scan through the previously mined FCCI to check for the existence and closeness of newly mined frequent colossal itemset.

Lemma 3. *An itemset $X \subseteq F_{final}$ during the feature enumeration, occurring in Y rows is closed iff the support of all the features in the IST_X is less than $|Y|$.*

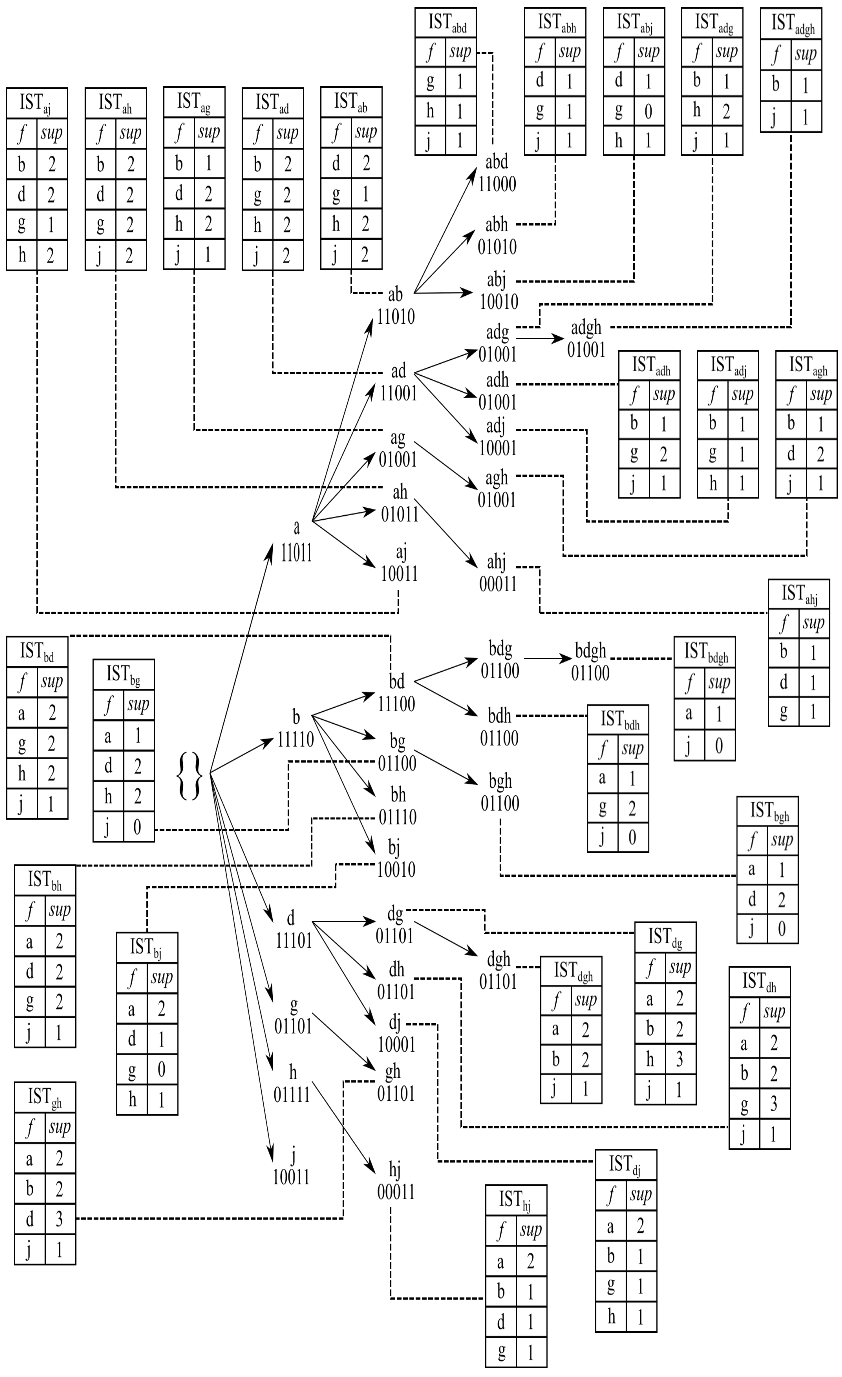

{}
a 11011
b 11110
d 11101
g 01101
h 01111
j 10011
ab 11010
ad 11001
ag 01001
ah 01011
aj 10011
abd 11000
abh 01010
abj 10010
adg 01001
adgh 01001
adh 01001
adj 10001
agh 01001
ahj 00011
bd 11100
bg 01100
bh 01110
bj 10010
bdg 01100
bdgh 01100
bdh 01100
bgh 01100
dg 01101
dh 01101
dj 10001
gh 01101
hj 00011
dgh 01101
IST_aj f sup b 2 d 2 g 1 h 2
IST_ah f sup b 2 d 2 g 2 j 2
IST_ag f sup b 1 d 2 h 2 j 1
IST_ad f sup b 2 g 2 h 2 j 2
IST_ab f sup d 2 g 1 h 2 j 2
IST_abd f sup g 1 h 1 j 1
IST_abh f sup d 1 g 1 j 1
IST_abj f sup d 1 g 0 h 1
IST_adg f sup b 1 h 2 j 1
IST_adgh f sup b 1 j 1
IST_adh f sup b 1 g 2 j 1
IST_adj f sup b 1 g 1 h 1
IST_agh f sup b 1 d 2 j 1
IST_ahj f sup b 1 d 1 g 1
IST_bd f sup a 2 g 2 h 2 j 1
IST_bg f sup a 1 d 2 h 2 j 0
IST_bdgh f sup a 1 j 0
IST_bdh f sup a 1 g 2 j 0
IST_bgh f sup a 1 d 2 j 0
IST_bh f sup a 2 d 2 g 2 j 1
IST_bj f sup a 2 d 1 g 0 h 1
IST_gh f sup a 2 b 2 d 3 j 1
IST_dgh f sup a 2 b 2 j 1
IST_dg f sup a 2 b 2 h 3 j 1
IST_dh f sup a 2 b 2 g 3 j 1
IST_dj f sup a 2 b 1 g 1 h 1
IST_hj f sup a 2 b 1 d 1 g 1

Figure 5.4. Bottom-Up Feature Enumerated Tree with Itemset Support Table for respective nodes

Proof. According to definition 8 (equations 1.1, 1.2, 1.3 and 1.4), if X is closed, then X=$f(Y)$. So it is necessary to prove that X=$f(Y)$ using IST_X. The IST_X*(feature)* provides the updated support value corresponding to the *feature* in IST_X. (For all)$\forall$ *feature* $\in$ (F_{final} - X), if IST_X*(feature)* $< |Y|$, then Y does not contain features from (F_{final} - X). This indicates that Y contains only an itemset X, thus proving X=$f(Y)$. Therefore, X is closed.

Lemma 4. *An itemset $X \subseteq F_{final}$ during the feature enumeration, occurring in Y rows is not closed iff the support of any one of the features in IST_X is equal to $|Y|$.*

Proof. According to definition 8 (equations 1.1, 1.2, 1.3 and 1.4), if X is not closed, then $X \neq f(Y)$. So it is necessary to prove that $X \neq f(Y)$ using IST_X. The IST_X*(feature)* provides the updated support value corresponding to the *feature* in IST_X. According to the steps followed for obtaining IST_X, the updated support for each *feature* in IST_X will never be greater than |Y|. (For any)$\forall$ *feature* $\in$ (F_{final} - X), if IST_X*(feature)* is equal to $|Y|$, then Y contain features from (F_{final} - X). Hence, $X \neq f(Y)$ is proved. Therefore, X is not closed.

The IST based closeness checking is based on Lemma 3 and Lemma 4. For Example, an itemset *ab* occurring in *124* during feature enumeration is closed. An itemset *ab* is closed because the support of all the features in IST_{ab} shown in Table 5.1a is less than |*124*|. An itemset *dg* occurring in *235* during feature enumeration is not closed. An itemset *dg* is not closed because the support of feature *h* in IST_{dg} shown in Table 5.1c is equal to |*235*|.

5.2.3 Proposed IST based Pruning Strategy

The IST based pruning strategy is proposed to efficiently cut down the feature enumerated search space. The proposed pruning strategy utilizes IST at every feature enumerated node as it provides prior information regarding the support of an itemset to be mined at descendant feature enumerated nodes without traversing them, unlike the existing FCCI mining algorithms which does not provide any prior information regarding the same.

Table 5.2. Itemset Support Table of feature enumerated *abd*, IST_{abd}

feature	sup
g	1
h	1
j	1

Given an itemset *X*, if the support of any *features* in IST_X is less than the *minsup*, then the descendant feature enumerated nodes corresponding to those *features* can be pruned as they do not contribute for the mining of frequent itemsets. For example, with *minsup* and *mincard* values set to 2, the IST_{abd} as shown in Table 5.2 gives prior information regarding the support of itemsets to be mined at descendant feature enumerated nodes (*abdg*, *abdh* and *abdj*). The support of *feature g*, *h* and *j* in IST_{abd} are less than *minsup*, which leads to the pruning of feature enumerated nodes *abdg*, *abdgh*, *abdghj*, *abdgj*, *abdh*, *abdhj* and *abdj* as they do not contribute for the mining of the frequent itemsets. The existing FCCI mining algorithms lack the ability to retrieve the prior information regarding the support of an itemset to be mined at descendant feature enumerated nodes, while the proposed pruning strategy overcomes this drawback.

5.2.4 Proposed Dynamic Switching Frequent Colossal Closed Itemset Mining (DSFCCIM) algorithm

The DSFCCIM algorithm is shown in Algorithm 5.1. Procedure 5.1a and Procedure 5.1b show the *RowEnum* procedure and *FeatureEnum* procedure respectively. The proposed DSFCCIM algorithm mines FCCI by performing a depth-first traversal of both row enumerated tree and feature enumerated tree. The preprocessed bitTable, *minsup*, and *mincard* are provided as an input to the algorithm. The FCCI, set of frequent colossal closed items is initialized to null. R'_{final} and F'_{final} are the set of rows and features to be enumerated respectively. The *rcomb* and *fcomb* are the first row and feature considered during the depth-first traversal of the row enumeration space and feature enumeration space respectively. The proposed DSFCCIM algorithm is enclosed with three switching conditions, which are as follows:

1. The switching condition at the beginning of the proposed DSFCCIM algorithm, as given in Equation 5.1.

2. The switching condition at row enumerated node during the row enumeration, as given in Equation 5.2.

3. The switching condition at feature enumerated node during the feature enumeration, as given in Equation 5.3.

The algorithm adopts either *RowEnum* procedure or *FeatureEnum* procedure depending on the switching condition. The switching condition in DSFCCIM algorithm is shown in Equation 5.1. ${}^{R_{final}}C_l$ is the number of different row enumerated node combinations of R_{final} distinct rows taken l at a time. ${}^{F_{final}}C_l$ is the number of different feature enumerated node combinations of F_{final} distinct features taken l at a time. The algorithm chooses the *RowEnum* procedure if the switching condition holds true else it will choose the *FeatureEnum* procedure.

$$\sum_{l=1}^{R_{final}} {}^{R_{final}}C_l \leq \sum_{l=1}^{F_{final}} {}^{F_{final}}C_l \tag{5.1}$$

Algorithm 5.1. DSFCCIM algorithm

```
Input: preprocessed bitTable, minsup, mincard.
Output: Set of Frequent Colossal Closed Itemsets,
        FCCI
   Initialisation: FCCI= $\emptyset$
                $R'_{final}$ = set of rows to be enumerated
                $F'_{final}$ = set of features to be enumerated
                rcomb = initial node in row enumeration
                fcomb = initial node in feature enumeration
                All bits in rbitset_result and fbitset_result
                initialized to 1
1: SwitchingCondition() //check for switch condition
2: if mining FCCI in row enumeration first
3:   RowEnum(rcomb,rbitset_result,$R'_{final}$,FCCI)
4: if mining FCCI in feature enumeration first
5:   FeatureEnum(fcomb,fbitset_result,$F'_{final}$,FCCI)
```

Procedure 5.1a. RowEnum(*rcomb*,rbitset_result,R'_{final},FCCI)

```
1:  Pruning 1: if node rcomb or its descendants does not reach till minsup
2:                 return
3:  calculate rbitset_result at the node rcomb
4:  Pruning 2: $\forall$ rid $\in R'_{final}$, if $RCT_{rcomb}(rid) < mincard$
5:                    delete rid from $R'_{final}$
6:  Optimization: if $|rcomb| < minsup$
7:                    discard closeness checking
8:                else
9:                    if Closeness_Checking(rcomb) == True
10:                       Add itemset mined at rcomb to FCCI
11: SwitchingCondition() //check for switch condition
12: if row enumeration
13:    for each row combination(rcomb) in $R'_{final}$
14:       RowEnum(rcomb,rbitset_result,$R'_{final}$,FCCI)
15: if switch to feature enumeration
16:    F"= itemset (mined at rcomb)
17:    for each feature enumeration(fcomb) in F"
18:       FeatureEnum(fcomb,fbitset_result,F",FCCI)
```

5.2.4.1 RowEnum Procedure

- **Pruning Strategy 1:** If the row enuemrated node *rcomb* or its descendants do not reach till *minsup*, then the *rcomb* and its descendants (if existing) are pruned to cut down the row enumerated search space. For example, the row enumerated nodes *46*, *5*, *56*, *6* shown in Figure 4.2 are pruned during the mining process, if the *minsup* value is set to 3.
- The rbitset_result is calculated at the row enumerated node *rcomb*. For example, the rbitset_result at row enumerated node *12* shown in Figure 4.2 is 11100000 (*abd*).
- **Pruning Strategy 2:** Pruning strategy 2 in *RowEnum* procedure highlight the proposed RCT based pruning strategy. The RCT based pruning strategy provides an added computational boost to the proposed DSFCCIM algorithm as it provides prior information regarding the cardinality of the itemsets to be mined.
- **Optimization:** If the number of *rid*s' in *rcomb* is less than *minsup*, then the closeness checking of the rowset *rcomb* is not required. For example, the closeness checking of all the 2-rowsets is not required when the *minsup* value is set to 3. The optimization in *RowEnum* procedure helps to skip closeness checking for (*minsup*-1) number of levels.

Switching condition is checked at every row enumerated node. The concept of the switching condition is to check the number of nodes to be traversed in the subtree at a row enumerated node. Row enumeration or feature enumeration is selected depending on a smaller number of nodes to be traversed in the subtree at a node. Let *m* be the number of rows to be enumerated at row enumerated node *rcomb* and *n* be the number of features in an itemset occurring at the row enumerated node *rcomb*, then the switching condition at the row enumerated node *rcomb* during row enumeration is shown in Equation 5.2. ${}^{m}C_l$ is the number of different row enumerated node combinations of *m* distinct rows taken *l* at a time. ${}^{n}C_l$ is the number of different feature enumerated node combinations of *n* distinct features taken *l* at a time. The algorithm switches to the feature enumeration space if the switching condition holds true, else it continues with a depth-first traversal of row enumeration space.

$$\sum_{l=1}^{n} {}^{n}C_l \leq \sum_{l=1}^{m} {}^{m}C_l \tag{5.2}$$

Procedure 5.1b. FeatureEnum(*fcomb*,fbitset_result,F'_{final},FCCI)

```
1:  Pruning 1: if node fcomb or its descendants does not reach till mincard
2:          return
3:  calculate fbitset_result at the node fcomb
4:  Pruning 2: $\forall$ feature $\in F'_{final}$, if $IST_{fcomb}$(feature)<minsup
5:              delete feature from $F'_{final}$
6:  Optimization: if |fcomb| <mincard
7:          discard closeness checking
8:         else
9:          if Closeness_Checking(itemset) == True
10:             Add itemset to FCCI
11: SwitchingCondition() //check for switch condition
12: if feature enumeration
13:   for each feature enumeration(fcomb) in $F'_{final}$
14:     FeatureEnum(fcomb,fbitset_result,$F'_{final}$,FCCI)
15: if switch to row enumeration
16:   R"= rowset (obtained at fcomb)
17:   for each row combination(rcomb) in R"
18:     RowEnum(rcomb,rbitset_result,R",FCCI)
```

5.2.4.2 FeatureEnum Procedure

- **Pruning Strategy 1:** If the feature enumerated node *fcomb* or its descendeants do not reach till *mincard*, then the *fcomb* and its descendants (if existing) are pruned to cut down the feature enumerated search space. For example, the feature enumerated nodes *gj*, *h*, *hj*, *j* as shown in Figure 5.2 are pruned during the mining process, if the *mincard* value is set to 3.

- The fbitset_result is calculated at the feature enumerated node *fcomb*. For example, the fbitset_result at feature enumerated node *ab* as shown in Figure 5.2 is 11010 (*124*).

- **Pruning Strategy 2:** Pruning strategy 2 in *FeatureEnum* procedure highlight the proposed IST based pruning strategy. The IST based pruning strategy provides an added computational boost to the proposed DSFCCIM algorithm as it provides prior information regarding the support of the itemsets to be mined.

- **Optimization:** If the number of features in *fcomb* is less than *mincard*, then the closeness checking of the itemset *fcomb* is not required. For example, the closeness checking of all the 2-itemsets is not required when the *mincard* value is set to 3. The optimization in *FeatureEnum* procedure helps to skip closeness checking for (*mincard*-1) number of levels.

Switching condition is checked at every feature enumerated node. The concept of the switching condition is to check the number of nodes to be traversed in the subtree at a feature enumerated node. Let *n* be the number of features to be enumerated at feature enumerated node *fcomb* and *m* be the number of rows in which an itemset *fcomb* has occurred, then the switching condition at feature enumerated node *fcomb* during feature enumeration is shown in Equation 5.3. ${}^{m}C_{l}$ is the number of different row enumerated node combinations of *m* distinct rows taken *l* at a time. ${}^{n}C_{l}$ is the number of different feature enumerated node combinations of *n* distinct features taken *l* at a time. The algorithm switches to the row enumeration space if the switching condition holds true, else it continues with depth-first traversal of feature enumeration space.

$$\sum_{l=1}^{m} {}^{m}C_{l} \leq \sum_{l=1}^{n} {}^{n}C_{l} \tag{5.3}$$

5.2.5 Complexity Analysis

For the dataset with a large number of rows and a large number of features, let R_{final} be the number of rows and F_{final} be the number of features after applying the proposed preprocessing technique. The space complexity of the bitTable is $\mathcal{O}(R_{final}F_{final})$. The rowset closeness checking method and pruning strategy of the proposed algorithm with the row enumeration approach take advantage of the RCT at the respective row enumerated node. During the row enumeration approach, the RCT at any particular row enumerated node *Y* will be in the memory until the completion of rowset closeness checking method and pruning strategy. Hence there will be only one RCT in the memory during the row enumeration approach and requires $\mathcal{O}(R_{final} - |Y|)$ to be in the memory. The space complexity during the row enumeration approach is $\mathcal{O}(R_{final}F_{final} + (R_{final} - |Y|))$. The itemset closeness checking method and pruning strategy of the proposed algorithm with the feature enumeration approach take advantage of the IST at the respective feature enumerated node. During the feature enumeration approach, the IST at any particular feature enumerated node *X* will be in the memory until the completion of itemset closeness checking method and pruning strategy. Hence there will be only one IST in the memory during the feature enumeration approach and requires $\mathcal{O}(F_{final} - |X|)$ to be in the memory. The space complexity during the feature enumeration approach is $\mathcal{O}(R_{final}F_{final} + (F_{final} - |X|))$.

The proposed algorithm with row enumeration approach traverse all the row enumerated nodes in the worst case. The total number of row enumerated nodes that need to be traversed in the worst case is *u*, u=$\sum_{l=1}^{R_{final}} {}^{R_{final}}C_l$. The time required for the RCT based rowset closeness checking method and pruning strategy is $\mathcal{O}(R_{final} - |Y|)$. The time complexity is $\mathcal{O}(u(R_{final} - |Y|))$ during the row enumeration approach. The proposed algorithm with feature enumeration approach traverse all the feature enumerated nodes in the worst case. The total number of feature enumerated nodes that need to be traversed in the worst case is v, v=$\sum_{l=1}^{F_{final}} {}^{F_{final}}C_l$. The time required for the IST based itemset closeness checking method and pruning strategy is $\mathcal{O}(F_{final} - |X|)$. The time complexity is $\mathcal{O}(v(F_{final} - |X|))$ during the feature enumeration approach.

Let the total number of row enumerated nodes that need to be traversed in average case be, *c*, such that c=$\sum_{l=1}^{k} {}^{k}C_l$, where *k* is the level in the row enumerated tree up to which all the mined itemsets are colossal; all the nodes that are present in the levels higher than *k* will not be traversed as these levels do not contain any colossal itemsets; $k << R_{final}$ and $c << u$. The time complexity is $\mathcal{O}(c(R_{final} - |Y|))$ during the row enumeration approach in the average case. Let the total number of feature enumerated nodes that need to be traversed in average case be, *d*, such that d=$\sum_{l=1}^{s} {}^{s}C_l$, where *s* is the level in the feature enumerated tree up to which all the mined itemsets are frequent; all the nodes that are present in the levels higher than *s* will not be traversed as these levels do not contain any frequent itemsets; $s << F_{final}$ and $d << v$. The time complexity is $\mathcal{O}(d(F_{final} - |X|))$ during the feature enumeration approach in the average case.

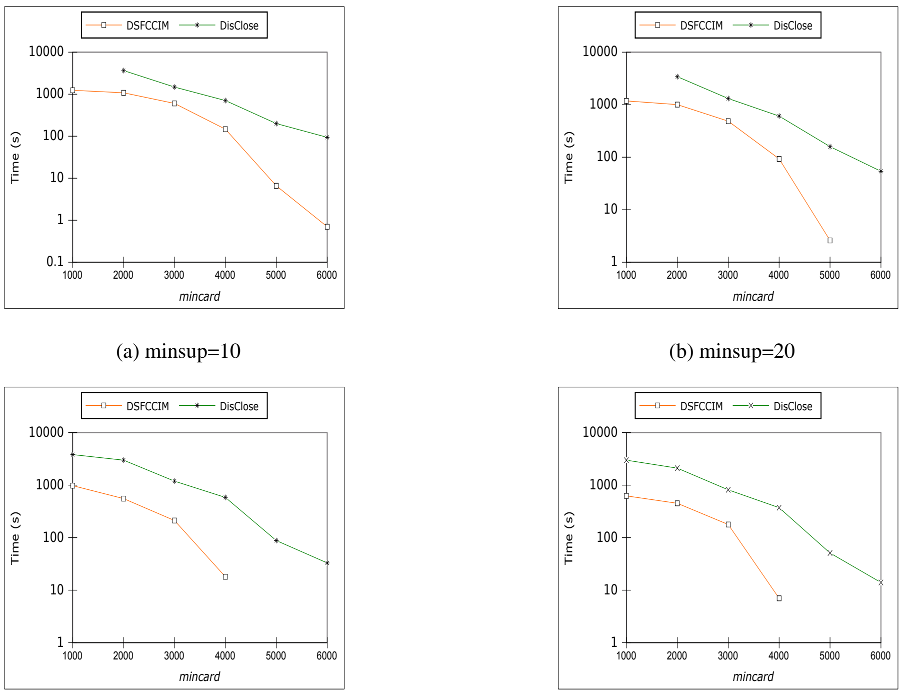
5.3 Results and Discussions

This section demonstrates the efficiency of the proposed DSFCCIM algorithm. The proposed DSFCCIM algorithm has been applied on ovarian cancer, lung cancer, prostate cancer, central nervous system embryonal tumor and Diffuse Large B-Cell Lymphoma (DLBCL) including Follicular Lymphoma datasets. The details of these datasets have been explained in section 3.3 of chapter 3. The proposed DSFCCIM algorithm has been compared with the DisClose algorithm. The DisClose algorithm outperforms the other existing FCCI mining algorithms. Hence it is chosen as representative for the experimental evaluation. The Pattern Fusion and BVBUC algorithm fail to mine the complete set of FCCI from the high dimensional dataset. Hence the Pattern Fusion and BVBUC algorithm cannot be considered for the experimental runtime evaluation. The proposed DSFCCIM algorithm and DisClose algorithm have been implemented in C++. The experiments were carried out on a computer with a specification of 3.4GHz core i7-3770

CPU, 8GB RAM, and 1TB hard disk.

Figure 5.5, Figure 5.6, Figure 5.7, Figure 5.8 and Figure 5.9 show the runtime comparison of proposed DSFCCIM algorithm with proposed EIP technique and DisClose algorithm at different values of *minsup* and *mincard* for ovarian cancer, lung cancer, prostate cancer, central nervous system embryonal tumor, and DLBCL (including Follicular Lymphoma) datasets respectively. The *x*-axis in Figure 5.5, Figure 5.6, Figure 5.7, Figure 5.8 and Figure 5.9 indicates different values of *mincard* and the *y*-axis indicates the runtime. Disclose algorithm was not able to record the runtime for ovarian cancer dataset when the (*minsup*, *mincard*) values were (10,1000) and (20,1000), as shown in Figure 5.5a and Figure 5.5b respectively.

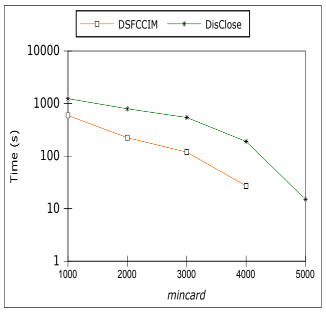
After applying the proposed EIP technique, the number of significant rows and significant features in the ovarian cancer dataset is zero when the *minsup* value reach 20, and *mincard* value reach 6000. This indicates that DSFCCIM algorithm is not obligatory to gauge the final result for ovarian cancer dataset when the *minsup* value reach 20, and *mincard* value reach 6000, as shown in Figure 5.5b. However, DisClose algorithm has to enumerate through huge row space to gauge the final result for ovarian cancer dataset when the *minsup* value reach 20, and *mincard* value reach 6000. Figure 5.6



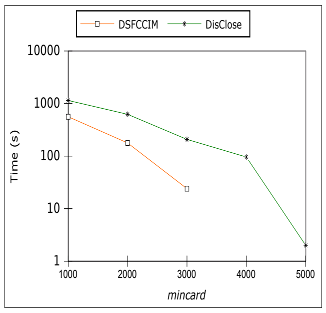
(a) minsup=10 (b) minsup=20

(c) minsup=30 (d) minsup=40

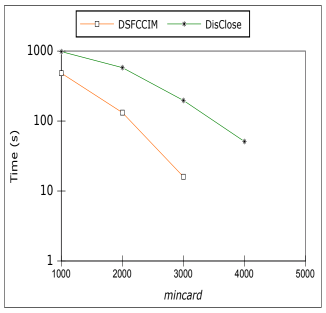
Figure 5.5. Runtime for Ovarian Cancer Dataset with proposed EIP technique for DSFCCIM algorithm



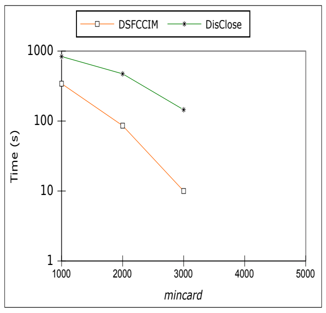
(a) minsup=10



(b) minsup=20

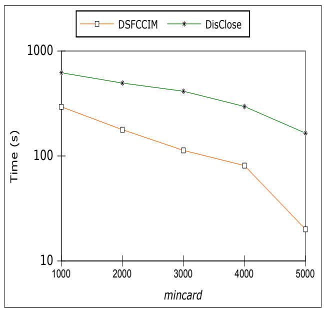


(c) minsup=30

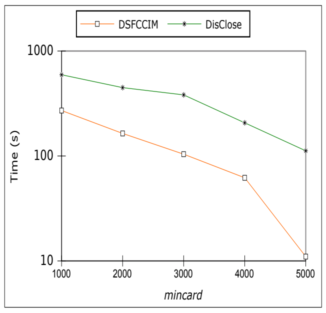


(d) minsup=40

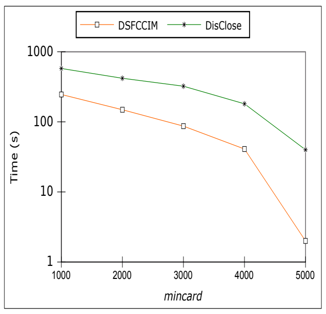
Figure 5.6. Runtime for Lung Cancer Dataset with proposed EIP technique for DSFCCIM algorithm



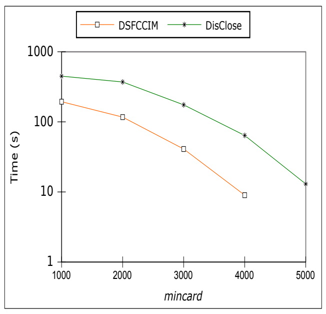
(a) minsup=10



(b) minsup=20

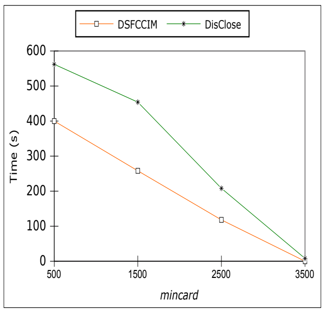


(c) minsup=30

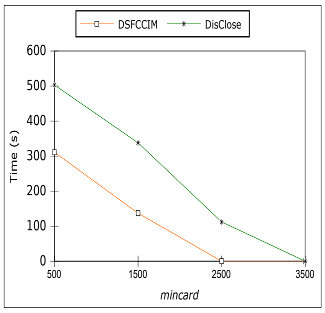


(d) minsup=40

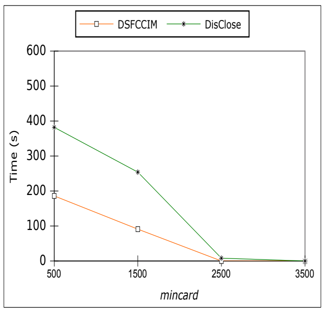
Figure 5.7. Runtime for Prostate Cancer Dataset with proposed EIP technique for DSFCCIM algorithm



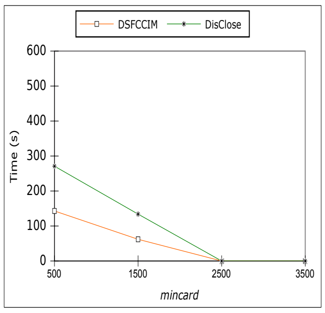
(a) minsup=5



(b) minsup=10

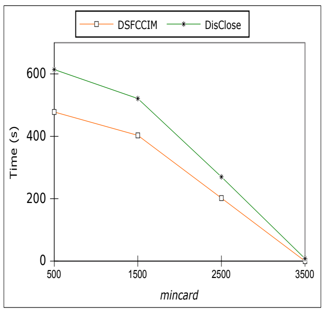


(c) minsup=15

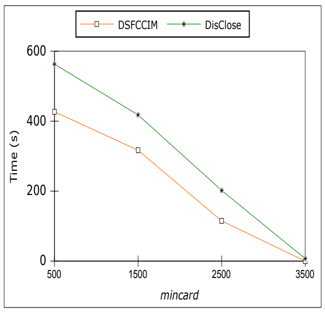


(d) minsup=20

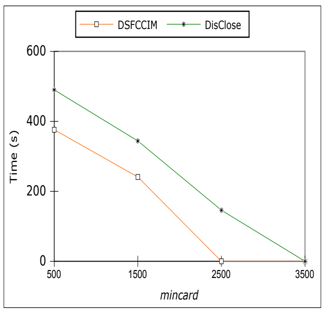
Figure 5.8. Runtime for Central Nervous System embryonal tumor Dataset with proposed EIP technique for DSFCCIM algorithm



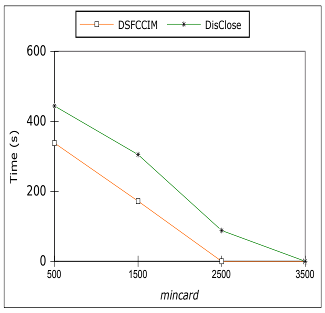
(a) minsup=5



(b) minsup=10



(c) minsup=15



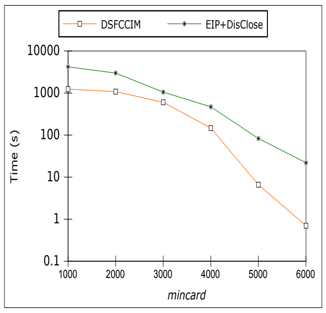
(d) minsup=20

Figure 5.9. Runtime for DLBCL (Including Follicular Lymphoma) Dataset with proposed EIP technique for DSFCCIM algorithm

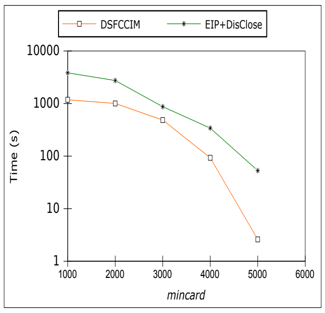
indicates that DSFCCIM algorithm is not obligatory to gauge the final result for lung cancer dataset when the *minsup* value reach 20, and *mincard* value reach 4000. However, Disclose is not obligatory to gauge the final result for lung cancer dataset when the *minsup* value reach 30, and *mincard* value reach 5000. Similar results have been observed for other experimental datasets for different values of *minsup* and *mincard*.

Figure 5.5, Figure 5.6, Figure 5.7, Figure 5.8, and Figure 5.9 show that DSFCCIM algorithm outperforms the Disclose algorithm in terms of runtime. The proposed DSFCCIM algorithm outperforms the DisClose algorithm by (2516, 2363, 2385, 1610) seconds for the ovarian cancer dataset, as shown in Figure 5.5, when the (*minsup*, *mincard*) values are set to (10, 2000), (20, 2000), (30, 2000) and (40, 2000) respectively. Similar results have been observed for different values of (*minsup*, *mincard*). The proposed DSFCCIM algorithm outperforms the DisClose algorithm by (291, 262, 251, 240) seconds for the prostate cancer dataset, as shown in Figure 5.7, when the (*minsup*, *mincard*) values are set to (10, 2000), (20, 2000), (30, 2000) and (40, 2000) respectively. Similar results have been observed for other experimental datasets for different values of (*minsup*, *mincard*). The results illustrate that DSFCCIM algorithm is efficient in handling the changing characteristics of data subset during the mining process. The results also illustrate the efficiency of proposed RCT and IST based closeness checking methods, and the efficiency of proposed RCT and IST based pruning strategies.

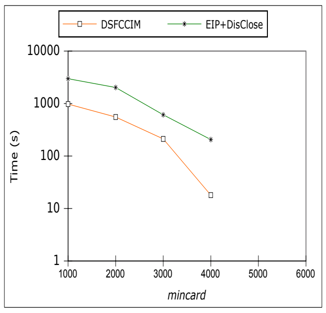
Figure 5.10, Figure 5.11, Figure 5.12, Figure 5.13, and Figure 5.14 show the runtime comparison of proposed DSFCCIM algorithm with proposed EIP technique and DisClose algorithm with proposed EIP technique at different values of *minsup* and *mincard* for ovarian cancer, lung cancer, prostate cancer, central nervous system embryonal tumor, and DLBCL (including Follicular Lymphoma) datasets respectively. The *x*-axis in Figure 5.10, Figure 5.11, Figure 5.12, Figure 5.13, and Figure 5.14 indicates different values of *mincard* and the *y*-axis indicates the runtime. After applying the proposed EIP technique to the proposed DSFCCIM algorithm and DisClose algorithm, it is evident that both DSFCCIM algorithm and DisClose algorithm are not obligatory to gauge the final result for ovarian cancer dataset when the *minsup* value reach 20, and *mincard* value reach 6000, as shown in Figure 5.10. After applying the proposed EIP technique to the proposed DSFCCIM algorithm and DisClose algorithm, it is evident that both DSFCCIM algorithm and DisClose algorithm are not obligatory to gauge the final result for lung cancer dataset when the *minsup* value reach 20, and *mincard* value reach 4000, as shown in Figure 5.11. Similar results have been observed for other experimen-



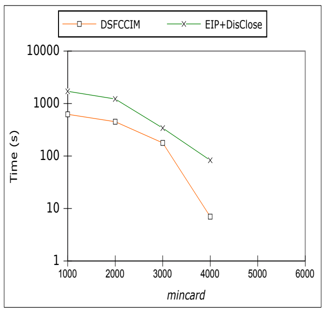
(a) minsup=10



(b) minsup=20

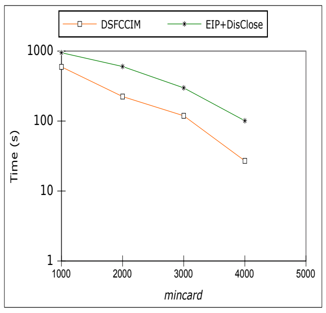


(c) minsup=30

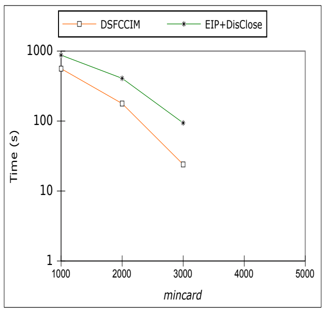


(d) minsup=40

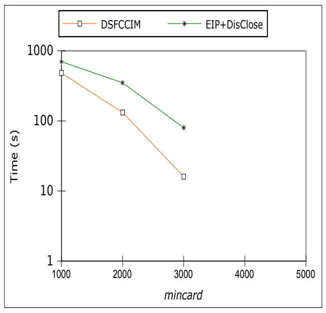
Figure 5.10. Runtime for Ovarian Cancer Dataset with proposed EIP technique for algorithms DSFCCIM and DisClose



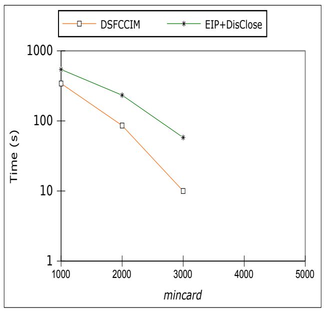
(a) minsup=10



(b) minsup=20

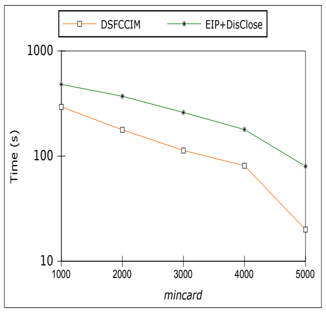


(c) minsup=30

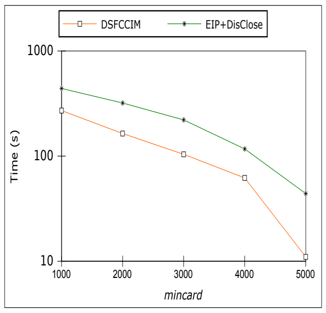


(d) minsup=40

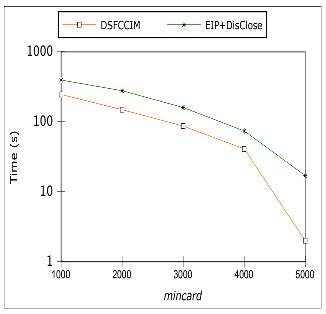
Figure 5.11. Runtime for Lung Cancer Dataset with proposed EIP technique for algorithms DSFCCIM and DisClose



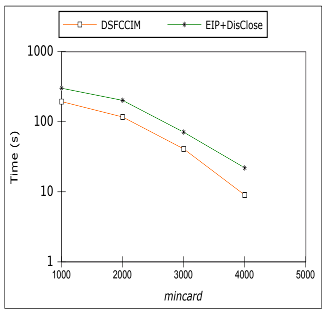
(a) minsup=10



(b) minsup=20



(c) minsup=30

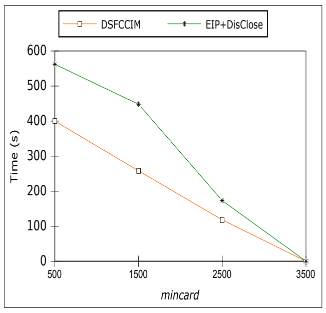


(d) minsup=40

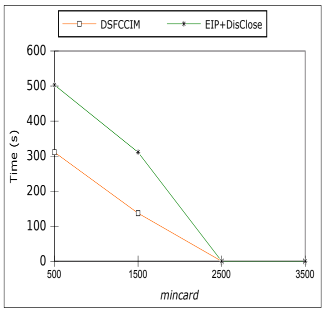
Figure 5.12. Runtime for Prostate Cancer Dataset with proposed EIP technique for algorithms DSFCCIM and DisClose

tal datasets for different values of *minsup* and *mincard*.

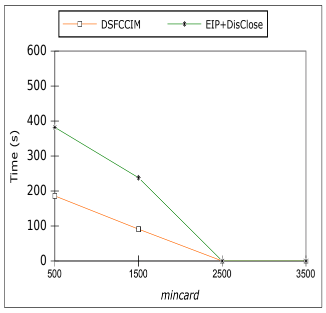
Figure 5.10, Figure 5.11, Figure 5.12, Figure 5.13, and Figure 5.14 show that DS-FCCIM algorithm with proposed EIP technique outperforms the Disclose algorithm with proposed EIP technique in terms of runtime. The proposed DSFCCIM algorithm with proposed EIP technique outperforms the DisClose algorithm with proposed EIP technique by (1850, 1688, 1420, 726) seconds for the ovarian cancer dataset, as shown in Figure 5.10, when the (*minsup*, *mincard*) values are set to (10, 2000), (20, 2000), (30, 2000) and (40, 2000) respectively. Similar results have been observed for different values of (*minsup*, *mincard*). The proposed DSFCCIM algorithm with proposed EIP technique outperforms the DisClose algorithm with proposed EIP technique by (167, 134, 110, 71) seconds for the prostate cancer dataset, as shown in Figure 5.12, when the (*minsup*, *mincard*) values are set to (10, 2000), (20, 2000), (30, 2000) and (40, 2000) respectively. Similar results have been observed for other experimental datasets for different values of (*minsup*, *mincard*). Figure 5.10, Figure 5.11, Figure 5.12, Figure 5.13, and Figure 5.14 illustrate the efficiency of proposed RCT and IST based closeness checking methods, and the efficiency of proposed RCT and IST based pruning strategies.



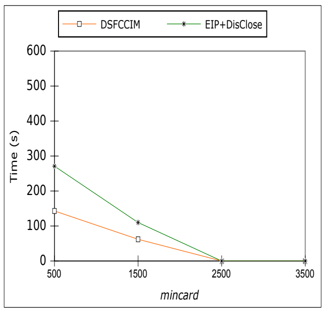
(a) minsup=5



(b) minsup=10

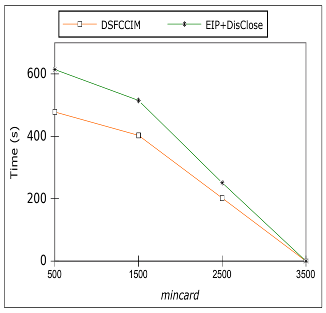


(c) minsup=15

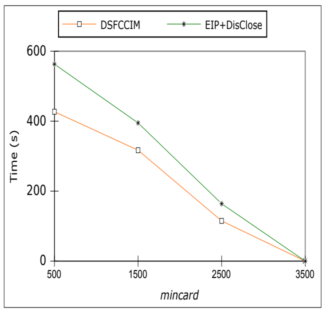


(d) minsup=20

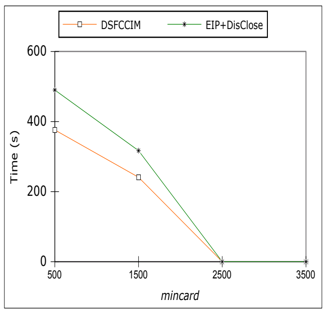
Figure 5.13. Runtime for Central Nervous System embryonal tumor Dataset with proposed EIP technique for algorithms DSFCCIM and DisClose



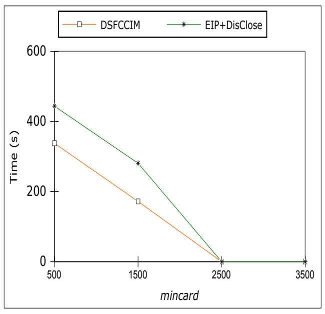
(a) minsup=5



(b) minsup=10



(c) minsup=15



(d) minsup=20

Figure 5.14. Runtime for DLBCL (Including Follicular Lymphoma) Dataset with proposed EIP technique for algorithms DSFCCIM and DisClose

The switching condition depends on the number of row enumerated nodes or feature enumerated nodes to be traversed in the subtree. The switching condition directs the DSFCCIM algorithm to start mining FCCI from all experimental datasets with the row enumeration approach, due to the data characteristics of the respective datasets. The characteristics of the data subset will change from one subset to another during the mining process. The changed characteristics of the data subset during the mining process decides the number of row enumerated nodes or feature enumerated nodes to be traversed in the subtree. It is observed that the DSFCCIM algorithm continues mining FCCI from all experimental datasets with row enumeration approach and do not switch to feature enumeration approach. The row enumeration approach is the best approach to continue mining FCCI from all experimental datasets because the number of row enumerated nodes to be traversed in the subtree at any point of the mining process is less than the number of feature enumerated nodes to be traversed.

5.4 Summary

In this chapter, Dynamic Switching Frequent Colossal Closed Itemset Mining (DSFCCIM) algorithm has been proposed to mine the FCCI from the dataset consisting of a large number of rows and a large number of features. The proposed DSFCCIM algorithm dynamically switches between bottom-up row enumerated tree and bottom-up feature enumerated tree to efficiently handle the changing characteristics of the data subset during the mining process. The proposed DSFCCIM algorithm is enclosed with RCT based closeness checking method and pruning strategy; it is also enclosed with IST based closeness checking method and pruning strategy. The experiments have been conducted on various datasets. The proposed DSFCCIM algorithm outperforms the DisClose algorithm by (2516, 2363, 2385, 1610) seconds for the ovarian cancer dataset, when the (*minsup*, *mincard*) values are set to (10, 2000), (20, 2000), (30, 2000) and (40, 2000) respectively. It is observed from the experiment results that the proposed DSFCCIM algorithm outperforms the existing DisClose algorithm in terms of runtime for all experimental datasets.

In the next chapter distributed and parallel mining of FCCI from the high dimensional dataset has been discussed.

Chapter 6

Distributed and Parallel Mining of Frequent Colossal Closed itemsets from the High Dimensional Dataset

The existing frequent colossal closed itemset mining algorithms for the high dimensional dataset are sequential and computationally expensive. Distributed and parallel computing is a good strategy to overcome the inefficiency of the existing sequential frequent colossal closed itemset mining algorithms. In this chapter, distributed and parallel algorithms have been proposed to efficiently mine FCCI from high dimensional datasets. The unbalanced intrinsic nature of the bottom-up row enumerated tree has been addressed by proposing the Balanced Distributed Parallel Frequent Colossal Closed Itemset Mining (BDPFCCIM) algorithm.

6.1 Distributed Row Enumerated Frequent Colossal Closed Itemset Mining (DREFCCIM) algorithm

The existing algorithms are computationally expensive and sequential in mining Frequent Colossal Closed itemsets (FCCI) from the high dimensional dataset. Distributed computing is a good strategy to overcome the inefficiency of the existing sequential algorithms. The Distributed Row Enumerated Frequent Colossal Closed Itemset Mining (DREFCCIM) algorithm has been proposed to overcome the drawbacks of the existing sequential algorithms. The proposed DREFCCIM algorithm mine the FCCI from the high dimensional dataset by traversing the bottom-up row enumerated tree due to the data characteristics of the high dimensional dataset.

The bottom-up row enumerated mining search space is efficiently cut down by the pruning strategy enclosed in the proposed DREFCCIM algorithm. The pruning strategy enclosed in the proposed DREFCCIM algorithm utilizes the Prune Table (PT) to efficiently cut down the row enumerated search space. The Prune Table and Prune Table based pruning strategy are explained in section 4.3 of chapter 4. The FCCI are mined by traversing the bottom-up row enumerated tree. The job of traversing the branches of the bottom-up row enumerated tree is distributed among the compute nodes. Figure 6.1 illustrates that the job of traversing the branches of the bottom-up row enumerated tree is distributed among six compute nodes.

Algorithm 6.1 shows the proposed DREFCCIM algorithm. The proposed DREFCCIM algorithm mine the FCCI by performing the depth-first traversal of bottom-up row

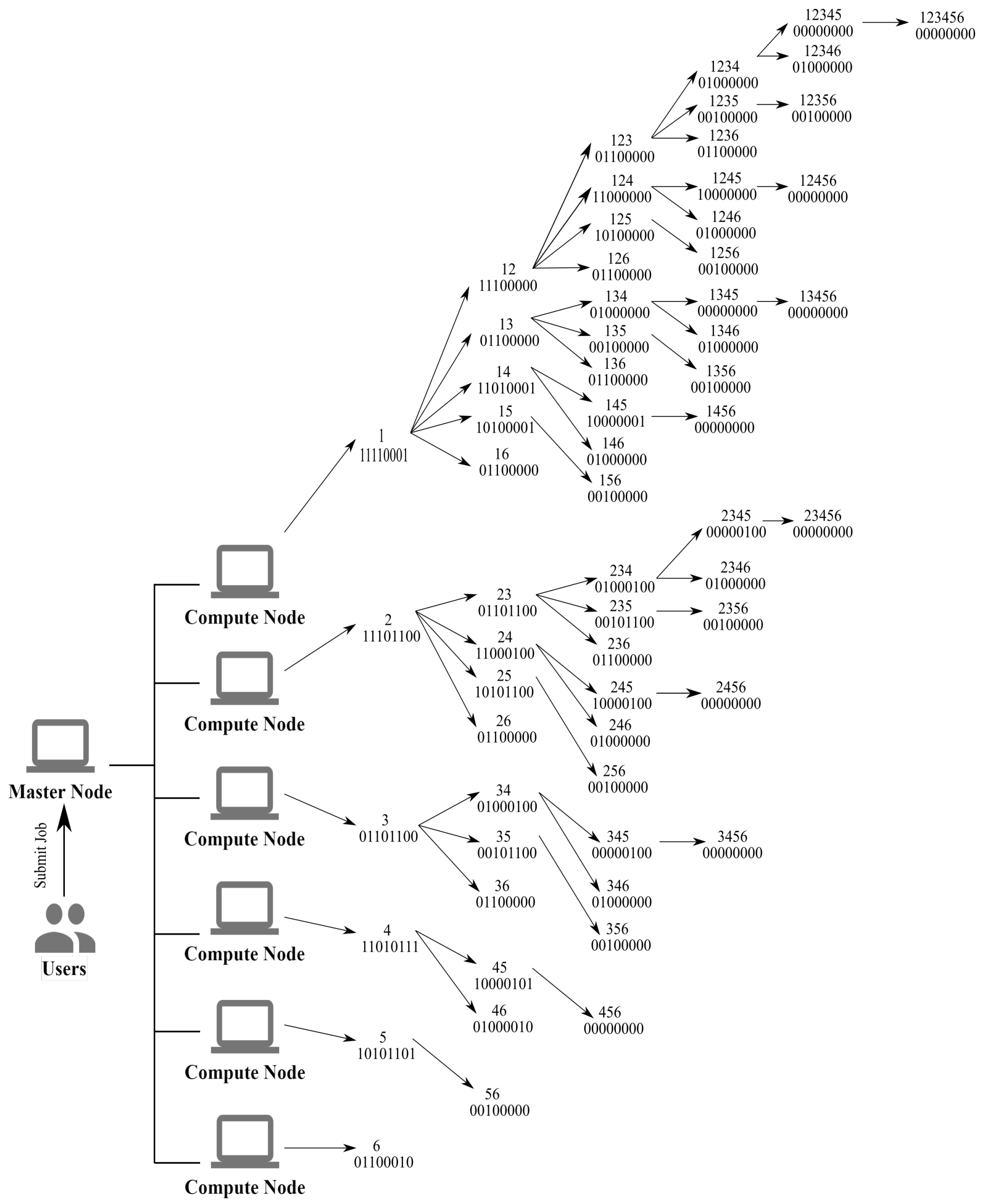


Figure 6.1. The job of traversing the branches of the bottom-up row enumerated tree is distributed among six compute nodes

enumerated tree. The preprocessed bitTable, minimum support threshold, and minimum cardinality threshold are provided as an input to the proposed algorithm. The proposed algorithm provides a set of frequent colossal closed itemsets as an output. The distributed approach of the proposed DREFCCIM algorithm is achieved by using the Message Passing Interface (MPI), which is the standardized message passing library.

The proposed algorithm initializes the MPI environment. The MPI communicator helps in identifying the number of compute nodes. Each compute node has a unique rank associated with the communicator, numbered from 0 to (n-1). The MPI communi-

Algorithm 6.1. DREFCCIM algorithm

```
Input: preprocessed bitTable, minsup, mincard.
Output: Set of Frequent Colossal Closed Itemsets, SFCCI
 1: Initializing MPI environment, MPI_Init(&argc, &argv)
 2: Number of Compute nodes, MPI_Comm_size(MPI_COMM_WORLD,&np)
 3: Rank of Compute nodes, MPI_Comm_rank(MPI_COMM_WORLD,&nrank)
 4: if(nrank==0)
 5:   Master Node
 6:     Scheduling of rid(s) among the compute nodes
 7:     Send rid(s) to the respective compute nodes
 8:     Receive mined FCCI from Compute Nodes
 9: else
10:   Compute Node
11:     Receive the respective rid(s) to be enumerated ($R'_{final}$)
12:     rcomb = initial row enumerated node
13:     Colossal_Closed(rcomb,bitset_result,$R'_{final}$,SFCCI)
14:     Send mined FCCI to Master Node
15: MPI_Finalize ( )
```

Procedure 6.1a. Colossal_Closed(*rcomb*,bitset_result,R'_{final},SFCCI))

```
1: if node rcomb does not reach till minsup
2:   return
3: calculate bitset_result at the node rcomb
4: Pruning: $\forall\ r_i \in R'_{final}$, if $PT_{rcomb}[r_i]$[card]<mincard
5:            delete $r_i$ from $R'_{final}$
6: checking the closeness of an itemsets, if closenesscheck(itemset)==True
7:   Add itemset to SFCCI
8:   for each row enumerated node(rcomb) in $R'_{final}$
9:     Colossal_Closed(rcomb,bitset_result,$R'_{final}$,SFCCI)
```

cator also helps in knowing the identity of the sender and receiver. The scheduling of the *rid*(s) (branches of the bottom-up row enumerated tree) among the compute nodes is done by the master node, as shown in Figure 6.1. The master node sends *rid*(s) to the respective compute nodes. The compute node receives the respective *rid*(s) to be enumerated from the master node. Each compute node traverse the assigned branches of the bottom-up row enumerated tree to mine FCCI.

The 'Colossal_Closed' procedure as shown in procedure 6.1a is invoked to mine the FCCI. The step 1 in 'Colossal_Closed' procedure indicates that, if the row enumerated node *rcomb* does not reach till *minsup* then, that row enumerated node and its subtree if exists is pruned to cut down the search space. The step 3 highlights about obtaining the bitset_result at the row enuemrated node *rcomb*. Step 4 indicates the proposed PT based pruning strategy. R'_{final} indicates the number of rows to be enumerated at a node *rcomb*, and if the cardinality of any of the rows in R'_{final} occurring in PT_{rcomb} is less than *mincard* then those rows are removed from R'_{final}, as they would not generate colossal itemsets. The proposed pruning strategy utilizes the Prune Table (PT) as it provides the prior information regarding the cardinality of the itemsets to be mined at the immediate child nodes without traversing them. Step 6 highlights the existing closeness checking of itemset obtained at row enumerated node *rcomb*. If the itemset satisfies the closeness checking then the itemset is added to the SFCCI. The algorithm continues with depth-first traversal of bottom-up row enumerated tree.

6.2 Parallel Row Enumerated Method for Mining Frequent Colossal Closed Itemsets from the High Dimensional Dataset

The existing preprocessing techniques are sequential and fail to prune the complete set of insignificant features and insignificant rows. The existing frequent colossal closed itemsets mining algorithms are sequential and computationally expensive. The proposed work highlights an Effective Improved Parallel Preprocessing (EIPP) technique to parallelly prune the complete set of insignificant features and insignificant rows from the high dimensional dataset and an efficient Parallel Frequent Colossal Closed Itemset Mining (PFCCIM) algorithm. The proposed PFCCIM algorithm is integrated with RCT based closeness checking method to check the closeness of a rowset and RCT based pruning strategy to cut down the row enumerated mining search space.

6.2.1 Effective Improved Parallel Preprocessing (EIPP) Technique

The proposed effective improved preprocessing technique parallelly prune the complete set of insignificant features and insignificant rows from the high dimensional dataset. The proposed Effective Improved Parallel Preprocessing (EIPP) technique incorporates the bitset approach. The EIPP technique is the parallel model of the proposed Effective Improved Preprocessing (EIP) technique, as explained in section 3.1 of chapter 3.

The proposed EIPP technique is divided into two tasks. First, Parallel Minimum Support Threshold Preprocessing (PMSTP) task to prune the insignificant features parallelly. Second, Parallel Minimum Cardinality Threshold Preprocessing (PMCTP) task to prune the insignificant rows parallelly. Algorithm 6.2 highlights the proposed EIPP technique. PMSTP task, as shown in Procedure 6.2a and PMCTP task, as shown in Procedure 6.2b are invoked by the proposed EIPP technique in an iterative manner un-

Algorithm 6.2. Effective Improved Parallel Preprocessing Technique

```
Input: bitTable, minsup, mincard.
Output: preprocessed bitTable
    Initialisation : P_flag = 1
 1: while (P_flag==1) do
 2:     PMSTP_task()
 3:     P_flag=0
 4:     PMCTP_task()
 5: end while
```

Procedure 6.2a. PMSTP_task()

```
 1: #pragma omp parallel for
 2: for (j=0;j<no_of_features;);j++ do
 3:     $support_j$=0
 4:     #pragma omp parallel for reduction (+:$support_j$)
 5:     for (i=0;i<no_of_rows;i++) do
 6:         if (bitTable[i][j]==1) then
 7:             $support_j$ = $support_j$ + 1
 8:         end if
 9:     end for
10:     if ($support_j$<minsup) then
11:         delete $j^{th}$ feature
12:     end if
13: end for
```

Procedure 6.2b. PMCTP_task()

```
 1: #pragma omp parallel for
 2: for (i=0;i<no_of_rows;) do
 3:     $cardinality_i$=0
 4:     #pragma omp parallel for reduction (+:$cardinality_j$)
 5:     for (j=0;j<no_of_features;j++) do
 6:         if (bitTable[i][j]==1) then
 7:             $cardinality_i$ = $cardinality_i$ + 1
 8:         end if
 9:     end for
10:     if ($cardinality_i$<mincard) then
11:         delete $i^{th}$ row
12:         P_flag=1
13:     end if
14: end for
```

til all the features and rows in the bitTable satisfy the criteria of *minsup* and *mincard*, respectively.

6.2.2 Parallel Frequent Colossal Closed Itemset Mining (PFCCIM) algorithm

The existing algorithms for mining frequent colossal closed itemsets from the high dimensional dataset are sequential and computationally expensive. To surmount the drawbacks, an efficient Parallel Frequent Colossal Closed Itemset Mining (PFCCIM) algorithm integrating the following proposed techniques has been designed.

- An efficient Rowset Cardinality Table (RCT) based closeness checking method was proposed to check the closeness of a rowset. The proposed method checks the closeness of newly mined frequent colossal itemsets irrespective of the previously mined FCCI. It is not required to store the complete set of previously mined FCCI in the main memory as the closeness checking of a rowset indicates the closeness of an itemset mined at that particular rowset. The advantage of not having a dependency to check the closeness of a rowset by the proposed RCT based closeness checking method will help to a great extent for designing the proposed parallel row enumerated algorithm for mining FCCI from the high dimensional dataset.

- An efficient RCT based pruning strategy was proposed to cut down the mining search space by efficient utilization of minimum cardinality threshold. The RCT at the row enumerated node provides prior information regarding the cardinality of the itemsets to be mined at descendant nodes without traversing them, unlike existing FCCI mining algorithms, which do not provide any prior information regarding the same.

The concept of RCT with examples has been discussed in subsection 4.4.1 of chapter 4. The RCT based closeness checking method to check the closeness of a rowset has been explained in subsection 4.4.2 of chapter 4. The RCT based pruning strategy to cut down the row enumerated search space has been explained with examples in the subsection 4.4.3 of chapter 4.

The proposed PFCCIM algorithm mine FCCI from the high dimensional dataset by utilizing the parallel bottom-up row enumeration approach as the large cardinality itemsets are present at the initial levels of the bottom-up row enumerated tree. Figure 6.2 shows the parallel bottom-up traversal of the row enumerated tree. Figure 6.2 shows that the team of threads is forked from the master thread for the parallel bottom-up

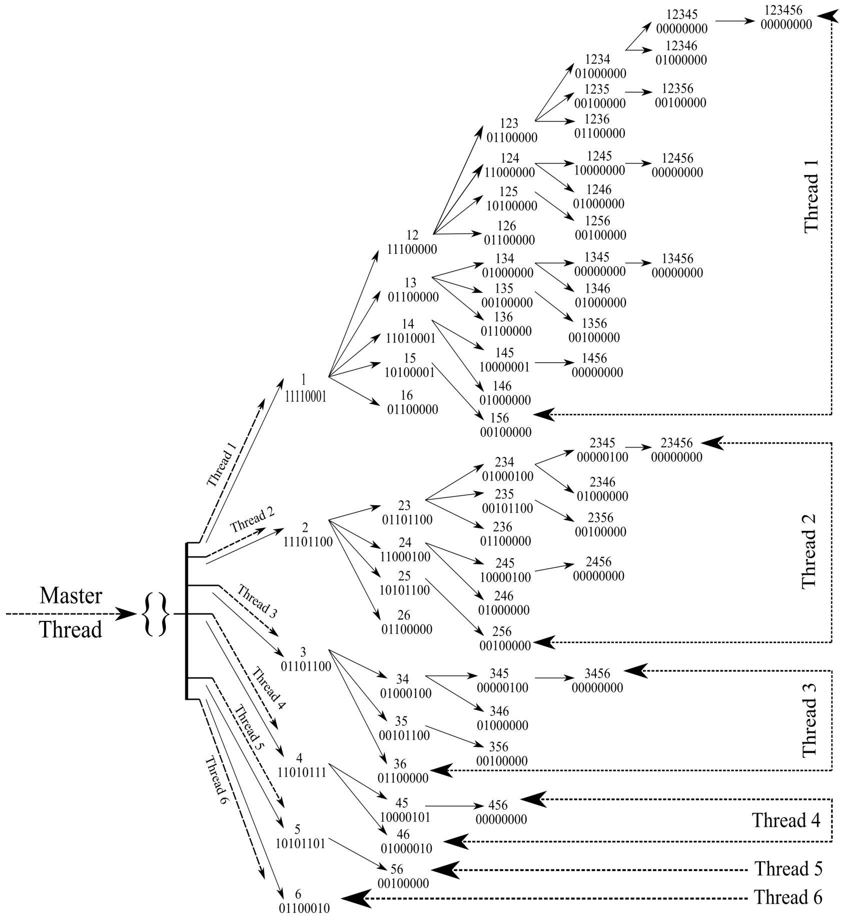


Figure 6.2. Parallel Bottom-Up Traversal of Row Enumerated Tree

traversal of the row enumerated tree. The row enumerated tree as shown in Figure 6.2 is constructed for the preprocessed bitTable shown in Table 3.5.

The FCCI are mined parallelly from the high dimensional dataset by PFCCIM algorithm. The PFCCIM algorithm is shown in Algorithm 6.3. The Procedure 6.3a highlights the PFCCIM procedure consisting of an efficient closeness checking method and an efficient pruning strategy. The PFCCIM algorithm mines the FCCI by performing the parallel depth-first traversal of the bottom-up row enumerated tree. The preprocesssed bitTable, *minsup*, and *mincard* are provided as input to the PFCCIM algorithm. The FCCI, set of frequent colossal closed itemsets is initialized to null. R''_{final} is the set of rows to be enumerated. The *rcomb* is the initial row enumerated node considered during the parallel depth-first traversal of the bottom-up row enumerated tree. The PFCCIM algorithm has been implemented using the Open Multi-Processing (OpenMP) application programming interface. The PFCCIM procedure is parallelly invoked by the multiple threads, which are forked from the master thread.

Algorithm 6.3. PFCCIM algorithm

```
Input: preprocessed bitTable, minsup, mincard.
Output: Set of Frequent Colossal Closed Itemsets,FCCI
    Initialisation: FCCI= $\emptyset$
    $R''_{final}$ = set of rows to be enumerated
    rcomb = initial node in row enumeration
1:  #pragma omp parallel
2:      All bit's in bitset_result are initialized to 1
3:      #pragma omp for
4:      for (i=rcomb; i<=no_of_rows; i++)
5:          PFCCIM(rcomb, bitset_result)
```

Procedure 6.3a. PFCCIM(*rcomb*, bitset_result)

```
1:  Pruning 1: if node rcomb or its descendants does not reach till minsup
2:              return
3:  calculate bitset_result at the node rcomb
4:  Pruning 2: $\forall$ rid $\in R''_{final}$, if $RCT_{rcomb}(rid) < mincard$
5:              delete rid from $R''_{final}$
6:  Optimization: if $|rcomb| < minsup$
7:                  discard closeness checking
8:              else
9:                  if Closeness_Checking(itemset,rcomb) == True
10:                     Add itemset to FCCI
11: for each row combination(r_combination) from rcomb
12:   PFCCIM(r_combination, bitset_result)
```

- **Pruning Strategy 1:** If the node *rcomb* or its descendants do not reach till *minsup* then, the *rcomb* and its descendants if existing, are pruned to cut down the search space. For example, the row enumerated node *6* shown in Figure 6.2 will be pruned during the mining process if the *minsup* value is set to 2, as it will not reach to the *minsup* level. The row enumerated nodes *46*, *5*, *56*, *6* shown Figure in 6.2 will be pruned during the mining process if the *minsup* value is set to 3.

Procedure 6.3b. Closeness_Checking(itemset,*rcomb*)

```
1:  Let $R_{final}$ be set of rows in preprocessed table
2:  (for any)$\forall$ rid $\in$ ($R_{final}$ - rcomb), if $RCT_{rcomb}(rid)$==card(itemset)
3:    flag_closed = false
4:    break
5:  if flag_closed == false
6:    return False
7:  else
8:    return True
```

- The bitset_result is calculated at node *rcomb*. For example, the bitset_result at row enumerated node *12* shown in Figure 6.2 is 11100000 (*abd*).

- **Pruning Strategy 2:** Pruning strategy 2 in the PFCCIM algorithm highlights the proposed RCT based pruning strategy. This strategy provides an added computational boost to the proposed PFCCIM algorithm as it provides prior information regarding the cardinality of the itemsets to be mined, whereas the existing FCCI mining algorithms do not provide prior information regarding the same.

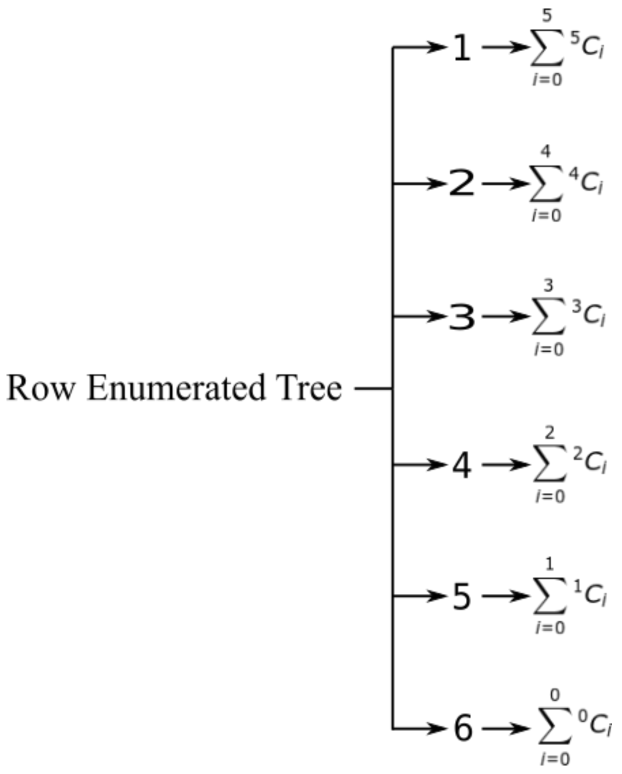
- **Optimization:** If the number of *rid*s' in *rcomb* is less than *minsup*, then the closeness checking of the rowset *rcomb* is not required. For example, the closeness checking of all the *2*-rowsets is not required when the *minsup* is set to 3. The optimization in PFCCIM helps to skip closeness checking for (*minsup*-1) number of levels.

- **Closeness Checking:** The procedure 6.3b and step 9 in procedure 6.3a highlight the proposed RCT based closeness checking method. The proposed RCT based closeness checking method is based on lemma 1 and lemma 2 as explained in subsection 4.4.2 of chapter 4. If the rowset satisfies the closeness checking, then the itemset mined from that rowset is added to FCCI. The algorithm continues with depth-first traversal of the bottom-up row enumeration space.

6.3 Distributed and Parallel Mining of Frequent Colossal Closed Itemsets with Load Balancing

The frequent colossal closed itemsets are mined from the high dimensional dataset by traversing the bottom-up row enumerated tree due to the data characteristics of the high dimensional dataset. The intrinsic nature of the row enumerated tree is typically unbalanced, as the number of nodes in each branch of row enumerated tree vary. It is important to properly distribute the branches of the row enumerated tree among the compute nodes to traverse it and mine the FCCI. The load of traversing the branches of row enumerated tree among the compute nodes should be balanced. The balanced distributed parallel algorithm has been designed to solve the inefficiency of the existing works. The Balanced Distributed Parallel Frequent Colossal Closed Itemset Mining (BDPFCCIM) algorithm has been proposed to mine FCCI from the high dimensional dataset. The EIPP technique prunes the complete set of insignificant features and insignificant rows. After applying the EIPP technique, let F_{final} be the set of significant features, R_{final} be the set of significant rows, n_{final} be the final number of significant features and m_{final} be the final number of significant rows.

The proposed BDPFCCIM algorithm efficiently distributes the branches of row enumerated tree to the compute nodes for traversing and mining FCCI. The branches of the row enumerated tree assigned to the compute nodes are traversed using the parallel bottom-up approach. The proposed algorithm is enclosed with an efficient RCT based rowset closeness checking method. If the closeness checking method indicates that the rowset is closed, then the itemset mined from that rowset is also closed. The proposed algorithm is also enclosed with the pruning strategy to efficiently cut down the nodes of the row enumerated tree, which do not produce FCCI. The pruning strategy utilizes RCT to get the prior information regarding the cardinality of the itemsets to be mined at the descendant nodes without traversing them. The concept of RCT with examples has been discussed in subsection 4.4.1 of chapter 4. The RCT based closeness checking method to check the closeness of a rowset has been explained in subsection 4.4.2 of chapter 4. The RCT based pruning strategy to cut down the row enumerated search space has been explained with examples in the subsection 4.4.3 of chapter 4.



Row Enumerated Tree
1 → $\sum_{i=0}^{5} {}^5C_i$
2 → $\sum_{i=0}^{4} {}^4C_i$
3 → $\sum_{i=0}^{3} {}^3C_i$
4 → $\sum_{i=0}^{2} {}^2C_i$
5 → $\sum_{i=0}^{1} {}^1C_i$
6 → $\sum_{i=0}^{0} {}^0C_i$

Figure 6.3. Number of nodes generated in each branch of row enumerated tree when m_{final} is 6

Figure 4.2 highlights that the intrinsic nature of the row enumerated tree is unbalanced. The number of row enumerated nodes that need to be traversed without pruning

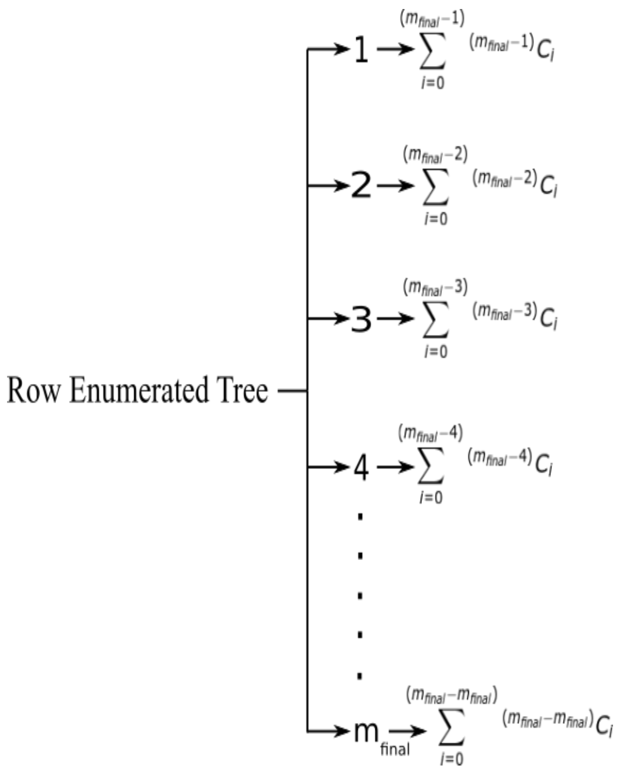

Row Enumerated Tree
1
2
3
4
m_{final}
$\sum_{i=0}^{(m_{final}-1)} {}^{(m_{final}-1)}C_i$
$\sum_{i=0}^{(m_{final}-2)} {}^{(m_{final}-2)}C_i$
$\sum_{i=0}^{(m_{final}-3)} {}^{(m_{final}-3)}C_i$
$\sum_{i=0}^{(m_{final}-4)} {}^{(m_{final}-4)}C_i$
$\sum_{i=0}^{(m_{final}-m_{final})} {}^{(m_{final}-m_{final})}C_i$

Figure 6.4. Number of nodes generated in each branch of the generalized row enumerated tree.

any of the row enumerated mining search space with respect to *mincard* is shown in Figure 4.2, and it is clear that the number of nodes in each branch of the row enumerated tree vary. Figure 6.3 shows the number of nodes generated in each branch of row enumerated tree when m_{final} is 6 and Figure 6.4 shows the number of nodes generated in each branch of the generalized row enumerated tree. It is important to properly distribute the branches of row enumerated tree among the compute nodes to mine the FCCI. The load of traversing the branches of row enumerated tree among the compute nodes should be balanced.

$$\sum_{i=0}^{5} {}^{5}C_i > \sum_{i=0}^{4} {}^{4}C_i + \sum_{i=0}^{3} {}^{3}C_i + \sum_{i=0}^{2} {}^{2}C_i + \sum_{i=0}^{1} {}^{1}C_i + \sum_{i=0}^{0} {}^{0}C_i \tag{6.1}$$

$$\sum_{i=0}^{4} {}^{4}C_i > \sum_{i=0}^{3} {}^{3}C_i + \sum_{i=0}^{2} {}^{2}C_i + \sum_{i=0}^{1} {}^{1}C_i + \sum_{i=0}^{0} {}^{0}C_i \tag{6.2}$$

The Equation 6.1 shows that the number of nodes generated from the first branch (rowset 1) of the row enumerated tree will be more than the collective number of nodes generated from the second branch (rowset 2) to the last branch (rowset 6) of the row

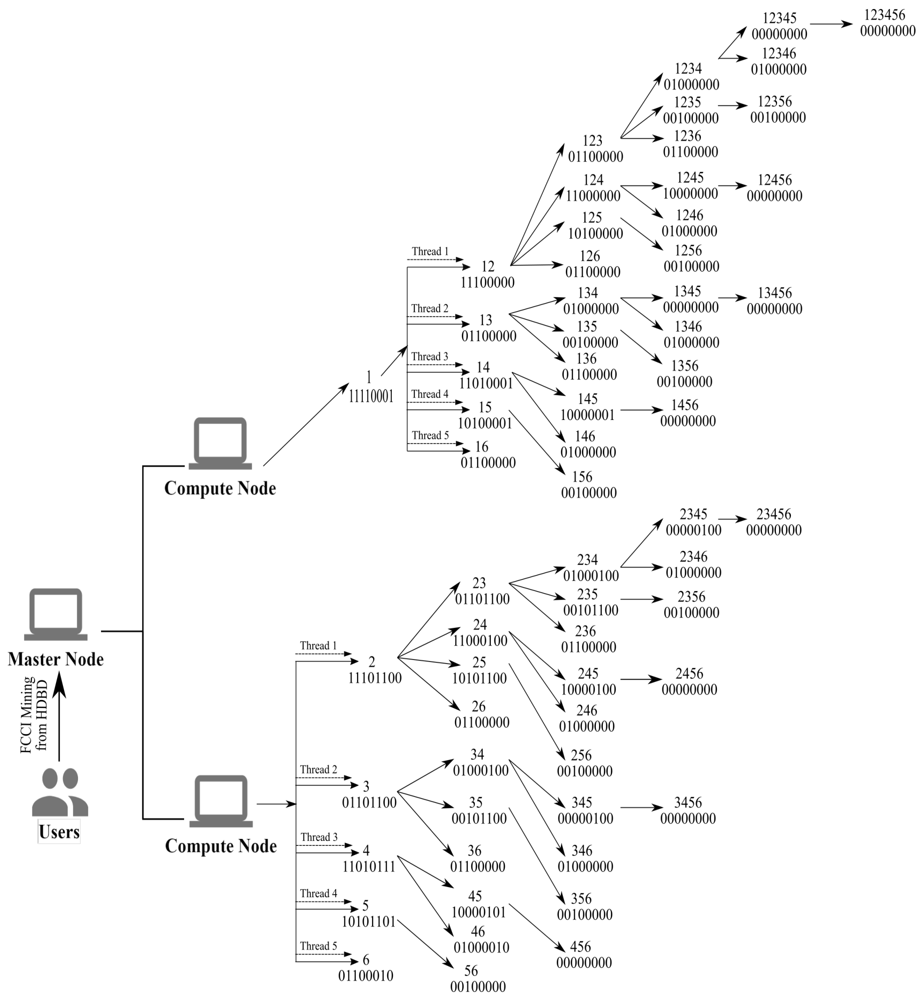


Figure 6.5. The branch distribution of row enumerated tree, when the available number of compute nodes are 2.

enumerated tree. The equation 6.2 shows that the number of nodes generated from the second branch (rowset 2) of the row enumerated tree is more than the collective number of nodes generated from the third branch (rowset 3) to the last branch (rowset 6) of the row enumerated tree. Let 'C' be the number of compute nodes available. The job of traversing the first branch (rowset 1) of row enumerated tree will be distributed to '$\frac{C}{2}$' number of compute nodes. The remaining '$\frac{C}{2}$' number of nodes will be assigned the job of traversing the remaining branches (second branch (rowset 2) to last branch (rowset n)) of the row enumerated tree. For example, if number of compute nodes available are 2, then the number of compute nodes assigned for traversing the first branch (rowset 1) of row enumerated tree will be one, and the number nodes assigned for traversing the second branch (rowset 2) to last branch (rowset 6) of row enumerated tree will be one.

Figure 6.5 shows the branch distribution of row enumerated tree, when the avail-

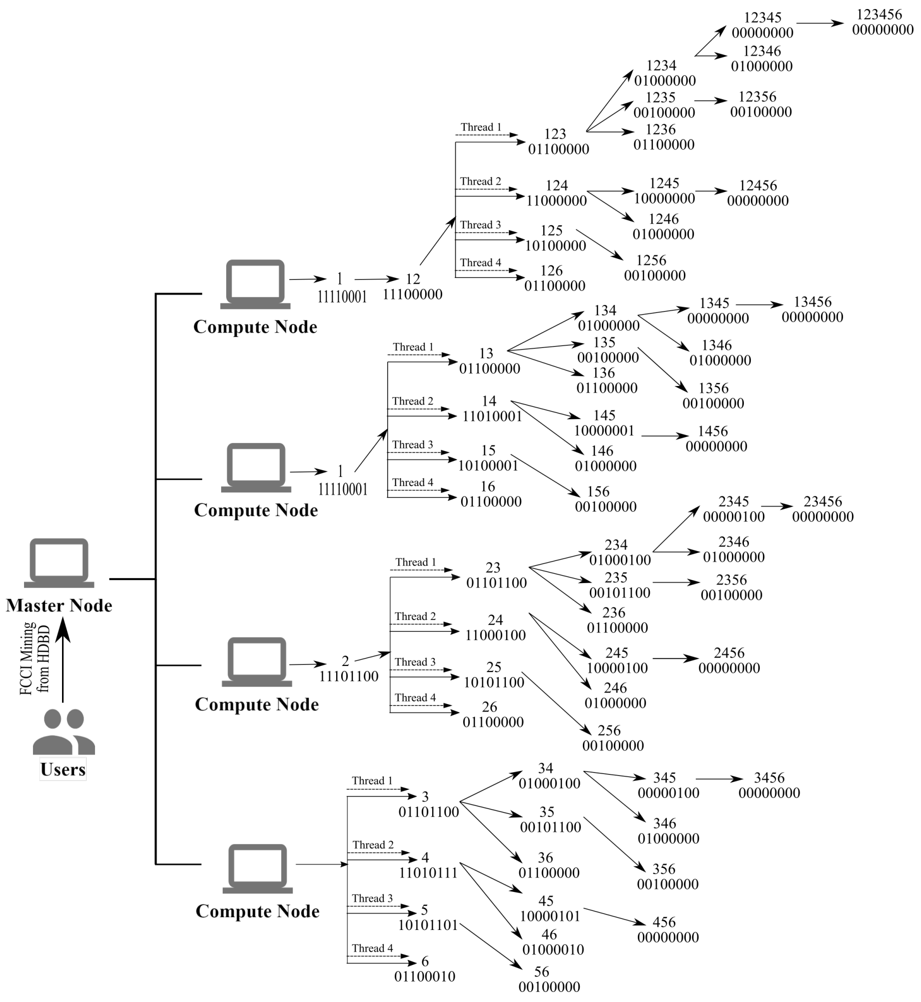


Figure 6.6. The branch distribution of row enumerated tree, when the available number of compute nodes are 4.

able number of compute nodes are 2. Figure 6.6 shows the branch distribution of row enumerated tree, when the available number of compute nodes are 4. The Figure 6.6 highlights that the first branch (rowset 1) of the row enumerated tree is distributed to two compute nodes and a second branch (rowset 2) to the last branch (rowset 6) of the row enumerated tree is distributed to two compute nodes. The traversal of the first branch (rowset 1) of the row enumerated tree should be distributed among the two assigned compute nodes with load balancing. The number of nodes generated by the first branch (rowset 12) of the rowset 1 will be more than the collective number of nodes generated from the second branch (rowset 13) to last branch (rowset 16) of the rowset 1. The load of traversing the branches should be balanced among the two assigned compute nodes. To balance the load among the compute nodes, the number of compute nodes assigned for traversing the first branch (rowset 12) of the rowset 1 will be one and number of compute nodes assigned for traversing the remaining branches (rowset 13 to rowset 16) of rowset 1 will be one, as shown in Figure 6.6.

The traversal of the second branch (rowset 2) to the last branch (rowset 6) of the row enumerated tree should be distributed among the two assigned compute nodes with load balancing. The number of nodes generated from the second branch (rowset 2) of the row enumerated tree is more than the collective number of nodes generated from third branch (rowset 3) to the last branch (rowset 6) of the row enumerated tree. The load of traversing the branches should be balanced among the two assigned compute nodes. To balance the load among the compute nodes, the number of compute nodes assigned to traverse the second branch (rowset 2) of the row enumerated tree will be one and number of compute nodes assigned to traverse the remaining branches (rowset 3 to rowset 6) of the row enumerated tree will be one as shown in Figure 6.6. The branches of the row enumerated tree assigned to the compute nodes are traversed using the parallel bottom-up approach as shown in Figure 6.5 and Figure 6.6. The branches of the row enumerated tree assigned to the compute nodes are traversed parallelly by the team of threads forked by the master thread.

Algorithm 6.4 shows the proposed Balanced Distributed Parallel Frequent Colossal Closed Itemset Mining (BDPFCCIM) algorithm. The rowset closeness checking method and an efficient pruning strategy are enclosed in BDPFCCIM procedure, as shown in Procedure 6.4a. The proposed BDPFCCIM algorithm distributes the branches of the row enumerated tree between the compute nodes with load balancing and then traverse the scheduled branches parallelly to mine the FCCI. The input for the proposed BDPFCCIM algorithm is the preprocessed bitTable, *mincard*, *minsup*. The mined FCCI from the high dimensional dataset is the output of the proposed algorithm. After applying the proposed EIPP technique, let F_{final} be the final set of significant features, R_{final} be the final set of significant rows, n_{final} be the final number of significant features and m_{final} be the final number of significant rows. The distributed approach of the proposed BDPFCCIM algorithm is achieved by using Message Passing Interface (MPI), the standardized message passing library and parallel approach of the proposed BDPFCCIM algorithms is achieved by using the Open Multi-Processing (OpenMP) application programming interface.

The proposed BDPFCCIM algorithm initializes the MPI environment. The MPI communicator helps in identifying the number of compute nodes. Each node has a unique rank associated with the communicator, numbered from 0 to (n-1). The MPI communicator also helps in knowing the identity of the sender and receiver. The

Algorithm 6.4. BDPFCCIM algorithm

Input: preprocessed bitTable, *mincard*, *minsup*.
Output: FCCI from High Dimensional Dataset

```
1:  $F_{final} \Rightarrow$ final set of significant features
2:  $R_{final} \Rightarrow$ final set of significant rows
3:  $n_{final} \Rightarrow$ final number of significant features
4:  $m_{final} \Rightarrow$ final number of significant rows
5:  Initializing MPI environment,
6:    MPI_Init(&argc, &argv)
7:  Number of Compute nodes,
8:    MPI_Comm_size(MPI_COMM_WORLD,&np)
9:  Rank of Compute nodes,
10:   MPI_Comm_rank(MPI_COMM_WORLD,&nrank)
11: if(nrank==0)
12:   Master Node
13:     Scheduling the branch traversal of row enumerated tree among compute nodes
      with load balancing. (Scheduling of rid(s) among compute nodes )
14:     Send rid(s) to respective compute nodes
15: else
16:   Compute Node
17:     Receive the respective rid(s) to be enumerated (R')
18:     ridcomb = initial row enumerated node
19:     All the bits' of bitset_result are set to 1.
20:     #pragma omp parallel for
21:       for(i=ridcomb; i $\leq$ no_of_rows (R'); i++)
22:         BDPFCCIM(ridcomb, bitset_result)
23:     Send mined FCCI to Master Node
24:   MPI_Finalize ( )
```

Procedure 6.4a. BDPFCCIM(*ridcomb*, bitset_result)

```
1:  $R''_{final} \Rightarrow$ set of rows to be enumerated
2:  Pruning 1: if node ridcomb or its descendants does not reach till minsup
3:                return
4:  calculate bitset_result at the node ridcomb
5:  Pruning 2: $\forall$ rid $\in R''_{final}$, if $RCT_{ridcomb}$(rid)<mincard
6:                delete rid from $R''_{final}$
7:  Optimization: if |ridcomb| <minsup
8:                  discard closeness checking
9:              else
10:                 if Closeness_Checking(itemset,ridcomb) == True
11:                   Add itemset to FCCI
12: for each row combination(r_combination) from rcomb
13:   BDPFCCIM(r_combination, bitset_result)
```

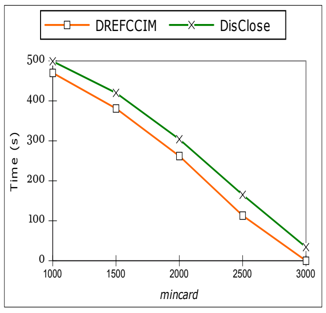
scheduling of branch traversal of the row enumerated tree between the compute nodes with load balancing is done by the master node. The master node sends *rid*(s) to the respective compute nodes. The compute node receives the respective *rid*(s) to be enumerated from the master node. Let *ridcomb* be the initial row to be enumerated. The compute nodes traverse the scheduled branch of the row enumerated tree in parallel manner by the team of threads, which are forked by the master thread. The procedure 6.4a highlighting the BDPFCCIM procedure is parallelly invoked by the team of threads. The BDPFCCIM procedure is enclosed with RCT based closeness checking method to check the closeness of the rowset and RCT based pruning strategy to cut down the row enumerated mining search space.

6.4 Results and Discussion

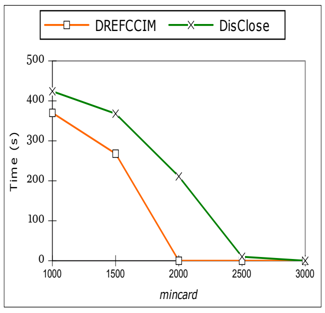
This section highlights the efficiency of the proposed distributed and parallel algorithms in mining frequent colossal closed itemsets from the high dimensional dataset. This section also highlights the speed-up of the distributed and parallel algorithms. The statistical significance analysis has also been discussed in this section.

6.4.1 Results of DREFCCIM Algorithm

This subsection emphasizes on the efficiency and speed-up of the proposed DREFCCIM algorithm. The proposed DREFCCIM algorithm has been applied on Diffuse Large B-Cell Lymphoma (DLBCL) and lung cancer test datasets. The details of the high dimensional datasets have been explained in section 3.3 of chapter 3. The proposed DREFCCIM algorithm has been compared with the DisClose algorithm. The DisClose algorithm outperforms the other existing FCCI mining algorithms. Hence it is chosen as representative for the experimental evaluation.

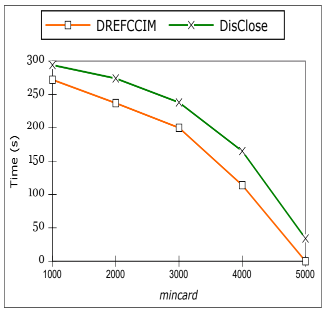


(a) minsup=5

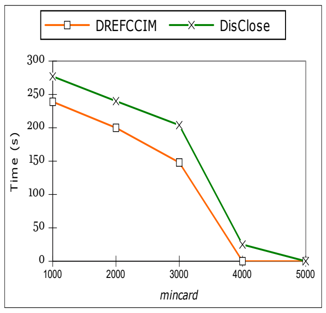


(b) minsup=15

Figure 6.7. Runtime of DREFCCIM (2 compute nodes) and DisClose for DLBCL Dataset



(a) minsup=5



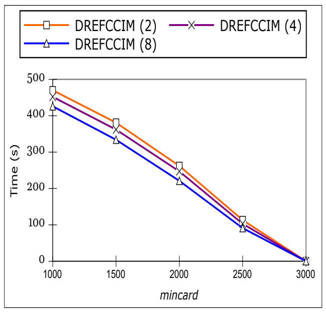
(b) minsup=15

Figure 6.8. Runtime of DREFCCIM (2 compute nodes) and DisClose for Lung Cancer Test Dataset

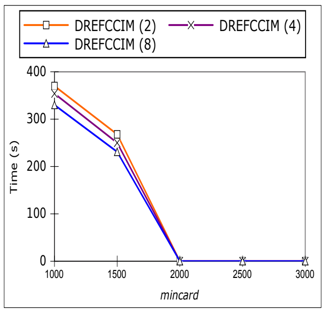
The experiments have been conducted on a cluster consisting of a master node and compute nodes. The master node and compute nodes have a specification of the Intel Xeon processor with 8GB RAM. The distributed computing of the proposed DREFCCIM algorithm has been achieved by the utilization of the Message Passing Interface (MPI) library.

Figure 6.7 and Figure 6.8 illustrate the runtime of the proposed DREFCCIM (2 compute nodes) algorithm and DisClose algorithm at different values of *minsup* and *mincard* for DLBCL and lung cancer test dataset, respectively. The *x*-axis in Figure 6.7 and Figure 6.8 indicates the different values of *mincard*. The y-axis in Figure 6.7 and Figure 6.8 illustrates the runtime. It has been observed from the experimental results that the runtime of the proposed DREFCCIM algorithm reduces as the *minsup* and *mincard* increases. The experimental results, as shown in Figure 6.7 and Figure 6.8, indicate that the proposed DREFCCIM algorithm is not obligatory to gauge the final result when the number of significant rows and significant features reach zero after applying the proposed EIPP technique for a given dataset, *minsup*, and *mincard*. Also, the experimental results highlight that the proposed DREFCCIM algorithm outperforms the Disclose algorithm in terms of runtime. The PT based pruning strategy enriched in the proposed DREFCCIM algorithm is efficient in cutting down the row enumerated search space. The distribution of traversing the branches of the row enumerated tree to the compute nodes by the proposed DREFCCIM algorithm makes it efficient compared to the DisClose algorithm in mining FCCI from the high dimensional dataset.

Figure 6.9 and Figure 6.10 show the runtime comparison between DREFCCIM (2 compute nodes), DREFCCIM (4 compute nodes), and DREFCCIM (8 compute nodes) at different values of *minsup* and *mincard* for DLBCL and lung cancer test dataset, respectively. The *x*-axis in Figure 6.9 and Figure 6.10 indicates the different values of

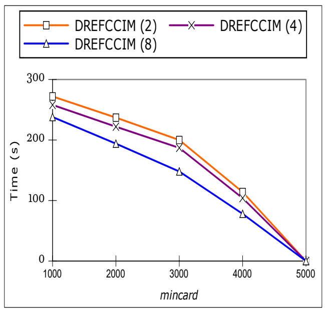


(a) minsup=5

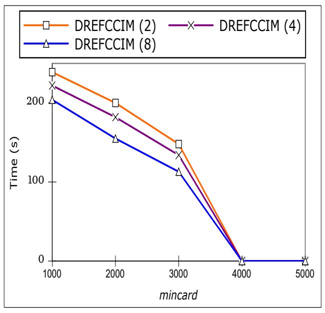


(b) minsup=15

Figure 6.9. Runtime of DREFCCIM (2 compute nodes), DREFCCIM (4 compute nodes) and DREFCCIM (8 compute nodes) for DLBCL Dataset



(a) minsup=5



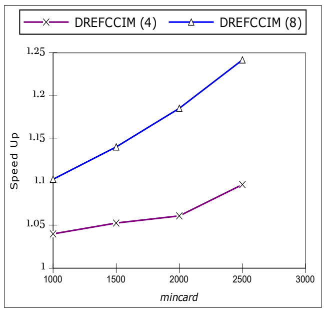
(b) minsup=15

Figure 6.10. Runtime of DREFCCIM (2 compute nodes), DREFCCIM (4 compute nodes) and DREFCCIM (8 compute nodes) for Lung Cancer Test Dataset

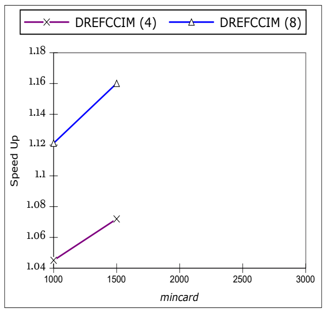
mincard. The *y*-axis in Figure 6.9 and Figure 6.10 indicates the runtime. The DREFCCIM algorithm with 8 compute nodes outperforms the DREFCCIM algorithm with 4 and 2 compute nodes in terms of runtime, as shown in Figure 6.9 and Figure 6.10.

The proposed DREFCCIM algorithm with 8 compute nodes outperforms the DisClose algorithm by (73, 94) seconds for the DLBCL dataset, as shown in Figure 6.7 and Figure 6.9, when the (*minsup*, *mincard*) values are set to (5, 1000) and (15, 1000) respectively. Similar results have been observed for different values of (*minsup*, *mincard*). The proposed DREFCCIM algorithm with 8 compute nodes outperforms the DisClose algorithm by (56, 73) seconds for the lung cancer test dataset, as shown in Figure 6.8 and Figure 6.10, when the (*minsup*, *mincard*) values are set to (5, 1000) and (15, 1000) respectively. Similar results have been observed for different values of (*minsup*, *mincard*).

Figure 6.11 and Figure 6.12 show the speed-up of the DREFCCIM (4 compute nodes) and DREFCCIM (8 compute nodes) with respect to the DREFCCIM (2 compute nodes) at different values of *minsup* and *mincard* for DLBCL and lung cancer test

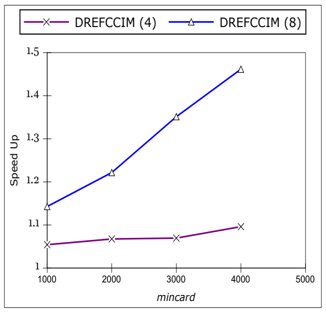


(a) minsup=5

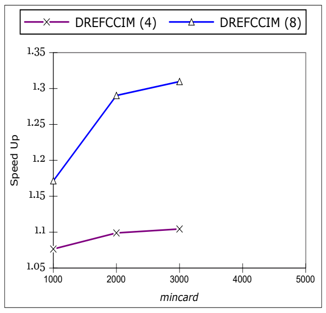


(b) minsup=15

Figure 6.11. Speedup of DREFCCIM (4 and 8 compute nodes) with respect to DREFCCIM (2 compute nodes) for DLBCL Dataset



(a) minsup=5



(b) minsup=15

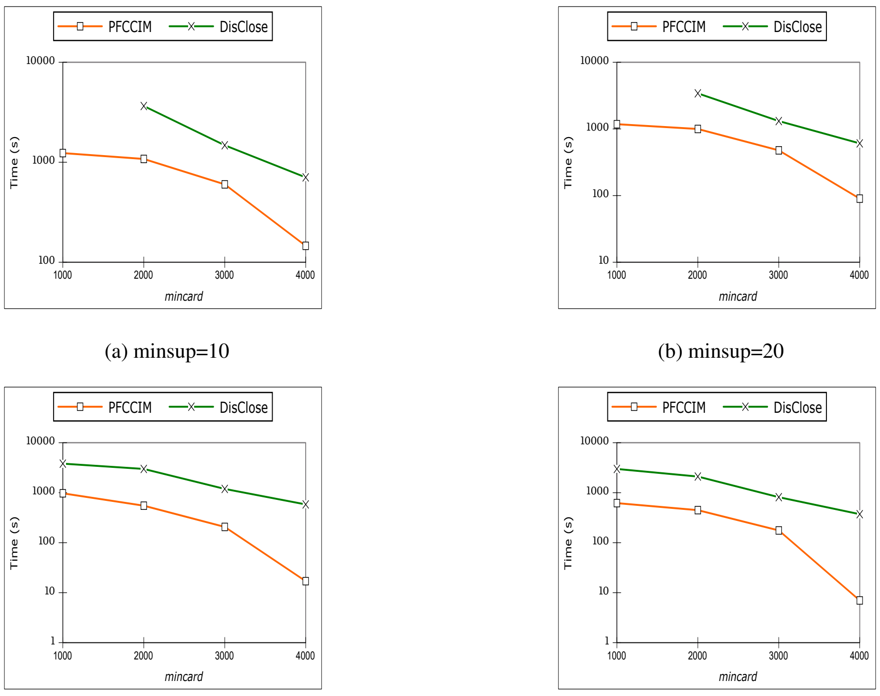
Figure 6.12. Speedup of DREFCCIM (4 and 8 compute nodes) with respect to DREFCCIM (2 compute nodes) for Lung Cancer Test Dataset

dataset, respectively. The x-axis in Figure 6.11 and Figure 6.12 indicates the different values of *mincard*. The y-axis in Figure 6.11 and Figure 6.12 indicates the speed-up. The proposed DREFCCIM algorithm (8 compute nodes) achieves the speed-up of (1.103, 1.121) with respect to the DREFCCIM algorithm (2 compute nodes) for DL-BCL dataset, as shown in Figure 6.11, when the (*minsup*, *mincard*) values are set to (5, 1000) and (15, 1000) respectively. Similar results have been observed for different values of (*minsup*, *mincard*). The proposed DREFCCIM algorithm (8 compute nodes) achieves the speed-up of (1.142, 1.171) with respect to the DREFCCIM algorithm (2 compute nodes) for lung cancer test dataset, as shown in Figure 6.12, when the (*minsup*, *mincard*) values are set to (5, 1000) and (15, 1000) respectively. Similar results have been observed for different values of (*minsup*, *mincard*).

The proposed DREFCCIM algorithm is not obligatory to gauge the final result for the DLBCL dataset when the *minsup* reaches 5 and *mincard* reaches 3000. This indicates that there is no speed-up factor when the (*minsup*, *mincard*) is (5, 3000), as shown in Figure 6.11a. The proposed DREFCCIM algorithm is not obligatory to gauge the final result for the DLBCL dataset when the *minsup* reaches 15 and *mincard* reaches 2000. This indicates that there is no speed-up factor when the (*minsup*, *mincard*) is (15, 2000), as shown in Figure 6.11b. Similar observations for the lung cancer test dataset have been made from Figure 6.12a and Figure 6.12b regarding the speed-up factor. The FCCI from the high dimensional dataset are mined by traversing the row enumerated tree, which by nature, as shown in Figure 4.2, exhibits the varying number of nodes in each branch of the row enumerated tree. This intrinsic nature of the row enumerated tree will lead to the average speed-up.

6.4.2 Results of PFCCIM Algorithm

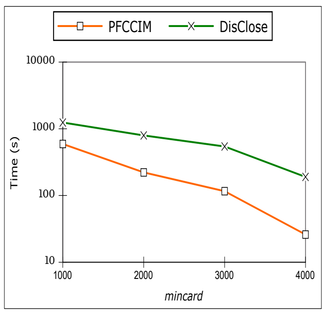
This section demonstrates the efficiency and speed-up of the proposed Parallel Frequent Colossal Closed Itemset Mining (PFCCIM) algorithm. The proposed PFCCIM algorithm has been applied on ovarian cancer, lung cancer, prostate cancer, central nervous system embryonal tumor and Diffuse Large B-Cell Lymphoma (DLBCL) including Follicular Lymphoma datasets. The details of these datasets have been explained in section 3.3 of chapter 3. The proposed PFCCIM algorithm has been compared with the DisClose algorithm. The DisClose algorithm outperforms the other existing FCCI mining



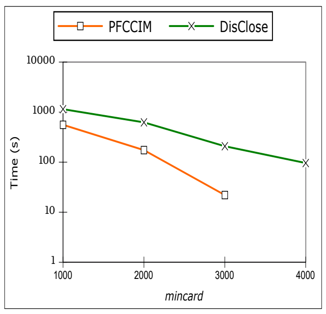
(a) minsup=10 (b) minsup=20

(c) minsup=30 (d) minsup=40

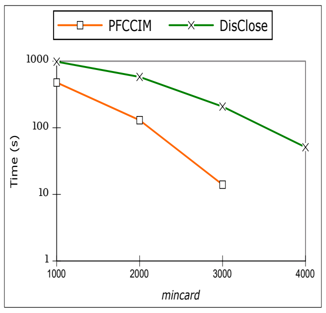
Figure 6.13. Runtime of PFCCIM (2 threads) and DisClose for Ovarian Cancer Dataset



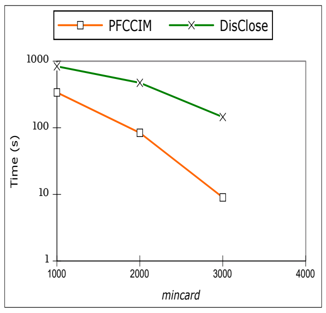
(a) minsup=10



(b) minsup=20



(c) minsup=30



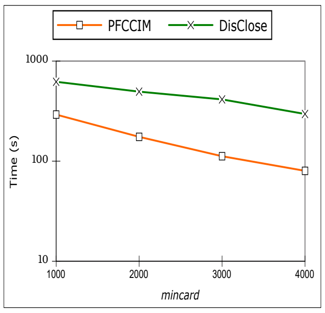
(d) minsup=40

Figure 6.14. Runtime of PFCCIM (2 threads) and DisClose for Lung Cancer Dataset

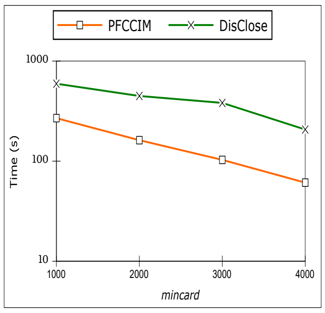
algorithms. Hence it is chosen as representative for the experimental evaluation. The parallelism of the proposed PFCCIM algorithm has been achieved by using the Open Multi-Processing (OpenMP) application programming interface. The experiments were carried out on a computer with a specification of 3.4GHz core i7-3770 CPU, 8GB RAM, and 1TB hard disk.

6.4.2.1 Runtime Analysis

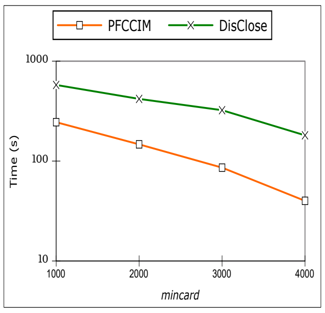
Figure 6.13, Figure 6.14, Figure 6.15, Figure 6.16, and Figure 6.17 show the runtime comparison of the proposed PFCCIM (2 threads) algorithm and the DisClose algorithm at different values of *minsup* and *mincard* for ovarian cancer, lung cancer, prostate cancer, central nervous system embryonal tumor, and DLBCL (including Follicular Lymphoma) datasets respectively. The x-axis in Figure 6.13 - Figure 6.17 indicates the different values of *mincard*, and the y-axis indicates the runtime. The Disclose algorithm was not able to record the runtime for the ovarian cancer dataset when the (*minsup*, *mincard*) was (10,1000) and (20,1000), as shown in Figure 6.13. After applying the proposed preprocessing technique, the number of significant rows and significant features in the ovarian cancer dataset is zero when the *minsup* reaches 20 and *mincard* reaches 6000. This indicates that the proposed PFCCIM algorithm is not obligatory to gauge the final result for the ovarian cancer dataset when the *minsup* reaches 20 and



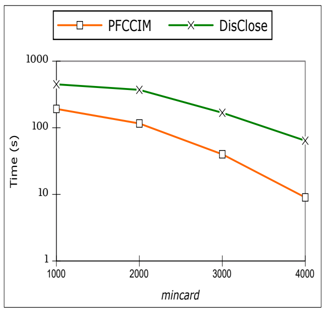
(a) minsup=10



(b) minsup=20

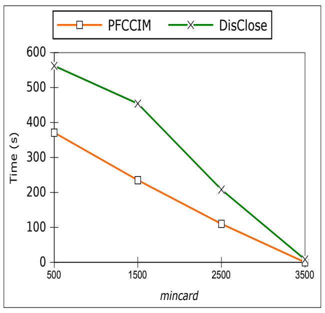


(c) minsup=30

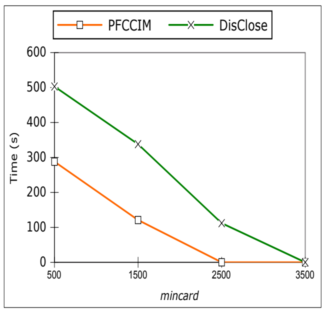


(d) minsup=40

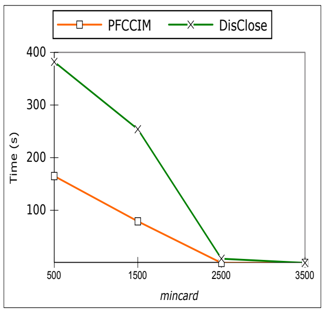
Figure 6.15. Runtime of PFCCIM (2 threads) and DisClose for Prostate Cancer Dataset



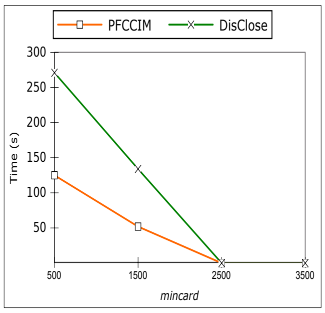
(a) minsup=5



(b) minsup=10

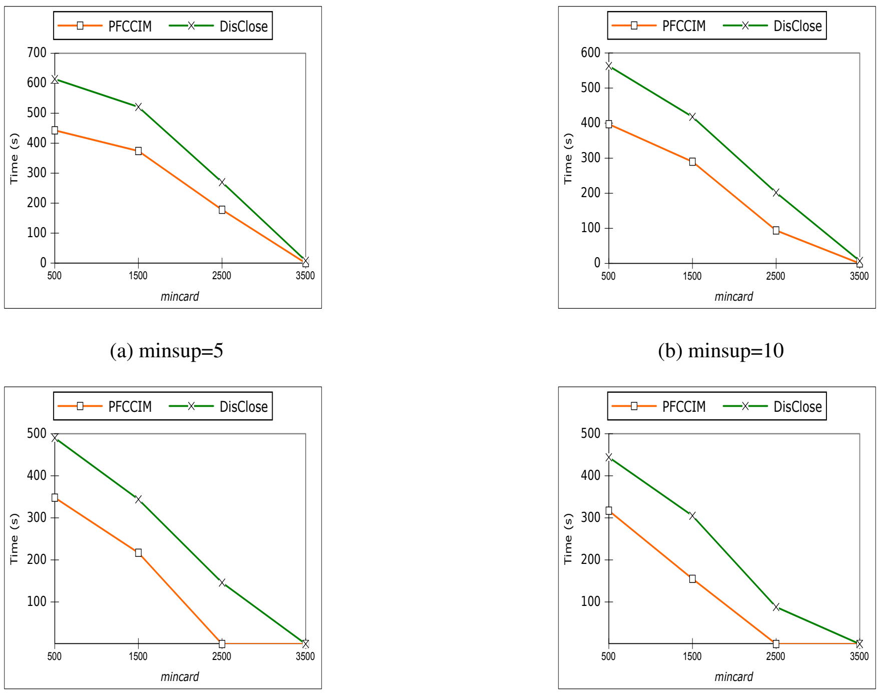


(c) minsup=15



(d) minsup=20

Figure 6.16. Runtime of PFCCIM (2 threads) and DisClose for Central Nervous System embryonal tumor Dataset



(a) minsup=5

(b) minsup=10

(c) minsup=15

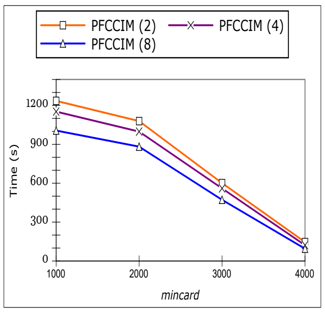
(d) minsup=20

Figure 6.17. Runtime of PFCCIM (2 threads) and DisClose for DLBCL (Including Follicular Lymphoma) Dataset

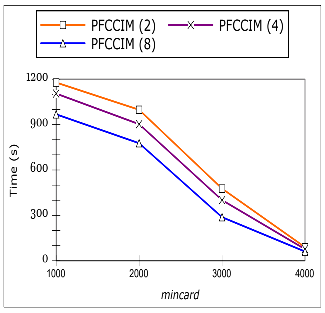
mincard reaches 6000, as shown in Figure 6.13b. However, the DisClose algorithm has to enumerate through a huge row enumerated space to gauge the final result for ovarian cancer dataset when the *minsup* value reach 20, and *mincard* value reach 6000. Figure 6.14 indicates that the proposed PFCCIM algorithm is not obligatory to gauge the final result for the lung cancer dataset when the *minsup* reaches 20 and *mincard* reaches 4000. Similar observations for the central nervous system embryonal tumor, and DLBCL (including the Follicular Lymphoma) datasets have been made from Figure 6.16 and Figure 6.17.

It can be observed from Figure 6.13 - Figure 6.17 that the runtime reduces as the *minsup* and *mincard* increases. The RCT based closeness checking method and pruning strategy enclosed with proposed PFCCIM algorithm help to efficiently mine the FCCI from the high dimensional dataset compared to DisClose algorithm. The proposed PFCCIM algorithm traverses the branches of the row enumerated tree in a parallel manner to mine the FCCI. Figure 6.13 - Figure 6.17 show that PFCCIM algorithm outperforms the Disclose algorithm in terms of runtime. The RCT based closeness checking method, RCT based pruning strategy and mining the FCCI parallelly helps the proposed PFCCIM algorithm in outperforming the DisClose algorithm.

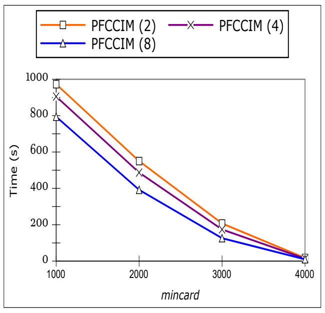
Figure 6.18, Figure 6.19, Figure 6.20, Figure 6.21 and Figure 6.22 highlight the runtime comparison of the proposed PFCCIM algorithm with the number of threads set to 2, 4, and 8 at different values of *minsup* and *mincard* for ovarian cancer, lung cancer, prostate cancer, central nervous system embryonal tumor, and DLBCL (including Follicular Lymphoma) datasets respectively. The *x*-axis in Figure 6.18 - Figure 6.22 indicates the different values of *mincard* and the *y*-axis indicates the runtime. Figure 6.18, Figure 6.19, Figure 6.20, Figure 6.21 and Figure 6.22 highlight that the proposed PFCCIM algorithm with 4 threads outperform the proposed PFCCIM algorithm with 2 threads in terms of runtime for ovarian cancer, lung cancer, prostate cancer, central nervous system embryonal tumor, and DLBCL (including Follicular Lymphoma) datasets respectively. It can also be observed that the proposed PFCCIM algorithm with 8 threads outperforms the proposed PFCCIM algorithm with 4 threads and 2 threads. The proposed PFCCIM algorithm with 8 threads outperforms the DisClose algorithm by (2646, 2450, 2597, 1794) seconds for the ovarian cancer dataset, as shown in Figure 6.13 and Figure 6.18, when the (*minsup*, *mincard*) values are set to (10, 2000), (20, 2000), (30, 2000) and (40, 2000) respectively. Similar results have been observed for different values of (*minsup*, *mincard*). The proposed PFCCIM algorithm with 8 threads outperforms the DisClose algorithm by (361, 321, 298, 282) seconds for the prostate



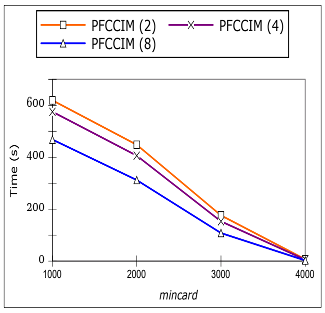
(a) minsup=10



(b) minsup=20

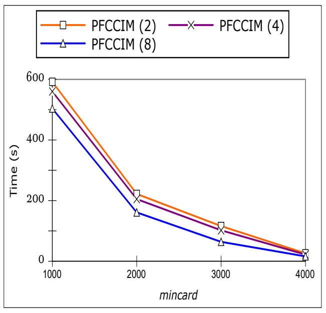


(c) minsup=30

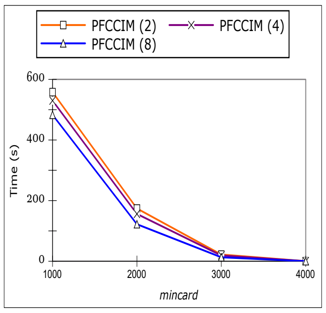


(d) minsup=40

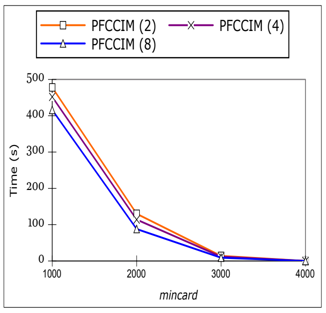
Figure 6.18. Runtime of PFCCIM (2 threads), PFCCIM (4 threads) and PFCCIM (8 threads) for Ovarian Cancer Dataset



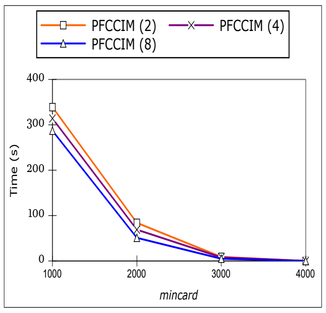
(a) minsup=10



(b) minsup=20

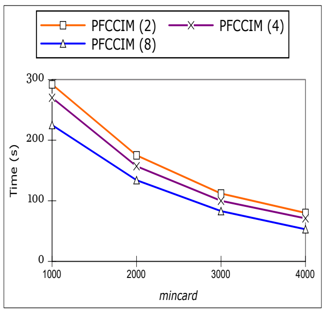


(c) minsup=30

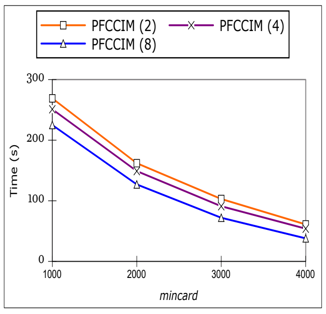


(d) minsup=40

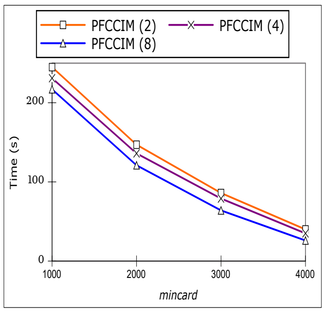
Figure 6.19. Runtime of PFCCIM (2 threads), PFCCIM (4 threads) and PFCCIM (8 threads) for Lung Cancer Dataset



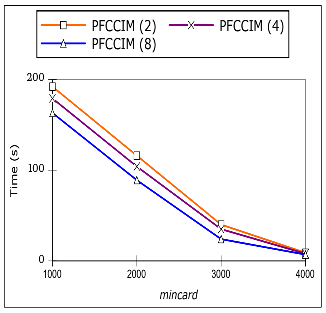
(a) minsup=10



(b) minsup=20

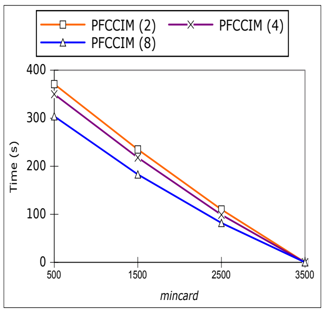


(c) minsup=30

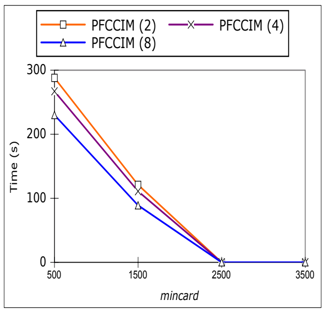


(d) minsup=40

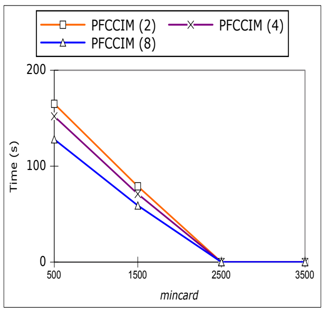
Figure 6.20. Runtime of PFCCIM (2 threads), PFCCIM (4 threads) and PFCCIM (8 threads) for Prostate Cancer Dataset



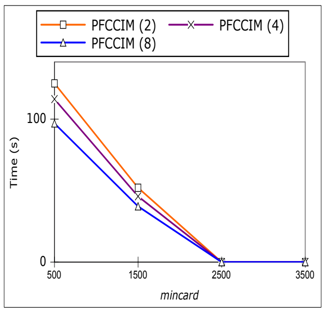
(a) minsup=5



(b) minsup=10

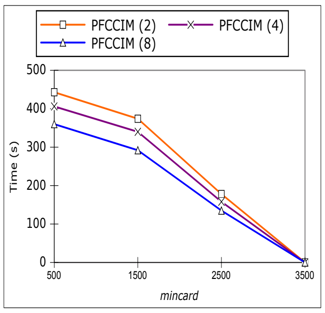


(c) minsup=15

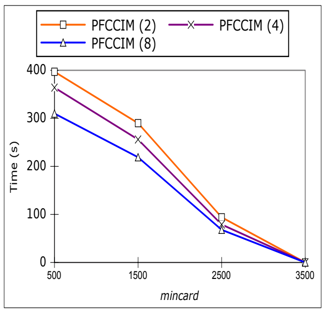


(d) minsup=20

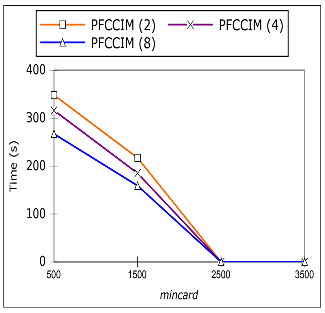
Figure 6.21. Runtime of PFCCIM (2 threads), PFCCIM (4 threads) and PFCCIM (8 threads) for Central Nervous System embryonal tumor Dataset



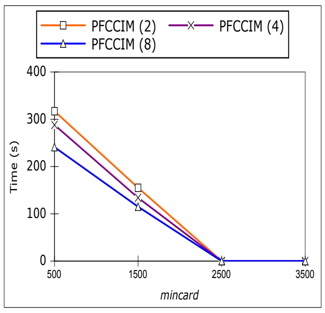
(a) minsup=5



(b) minsup=10



(c) minsup=15



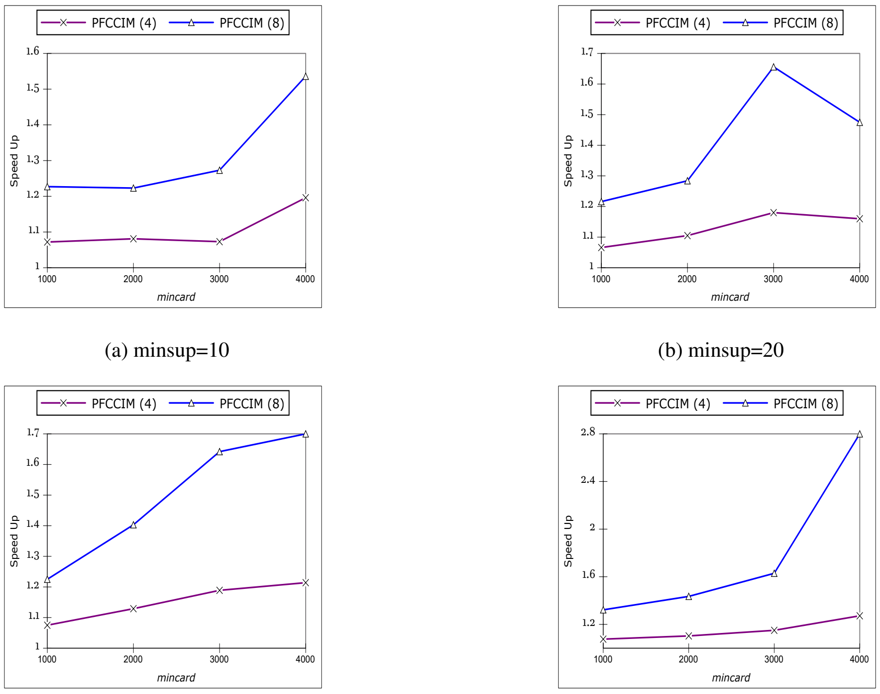
(d) minsup=20

Figure 6.22. Runtime of PFCCIM (2 threads), PFCCIM (4 threads) and PFCCIM (8 threads) for DLBCL (Including Follicular Lymphoma) Dataset

cancer dataset, as shown in Figure 6.15 and Figure 6.20, when the (*minsup*, *mincard*) values are set to (10, 2000), (20, 2000), (30, 2000) and (40, 2000) respectively. Similar results have been observed for other experimental datasets for different values of (*minsup*, *mincard*).

6.4.2.2 Speed-up Analysis

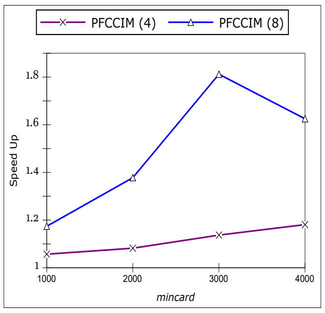
Figure 6.23, Figure 6.24, Figure 6.25, Figure 6.26, and Figure 6.27 show the speedup of the proposed PFCCIM algorithm (4 threads and 8 threads) with respect to the proposed PFCCIM algorithm (2 threads) at different values of *minsup* and *mincard* for ovarian cancer, lung cancer, prostate cancer, central nervous system embryonal tumor, and DLBCL (including the Follicular Lymphoma) datasets respectively. The *x*-axis and *y*-axis in Figure 6.23 - Figure 6.27 indicate the different values of *mincard* and the speed-up respectively. The proposed PFCCIM algorithm (8 threads) achieves the speed-up of (1.223, 1.284, 1.403, 1.435) with respect to the PFCCIM algorithm (2 threads) for the ovarian cancer dataset, as shown in Figure 6.23, when the (*minsup*, *mincard*) values are set to (10, 2000), (20, 2000), (30, 2000) and (40, 2000) respectively. Similar results have been observed for different values of (*minsup*, *mincard*). The proposed PFCCIM algorithm (8 threads) achieves the speed-up of (1.305, 1.275, 1.214, 1.303) with respect



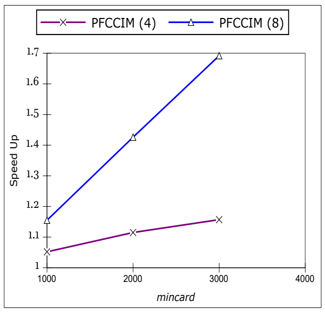
(a) minsup=10 (b) minsup=20

(c) minsup=30 (d) minsup=40

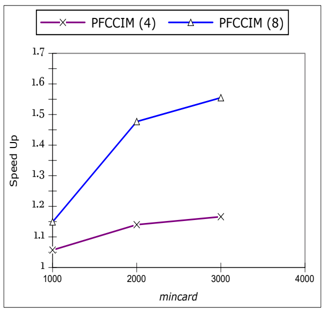
Figure 6.23. Speedup of PFCCIM (4 threads and 8 threads) with respect to PFCCIM (2 threads) for Ovarian Cancer Dataset



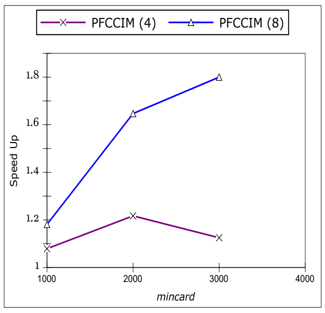
(a) minsup=10



(b) minsup=20

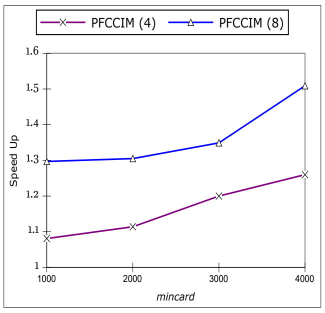


(c) minsup=30

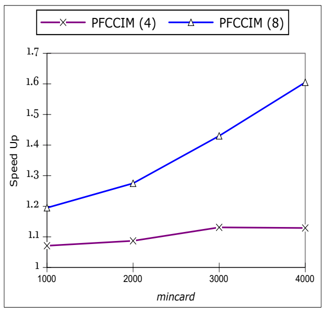


(d) minsup=40

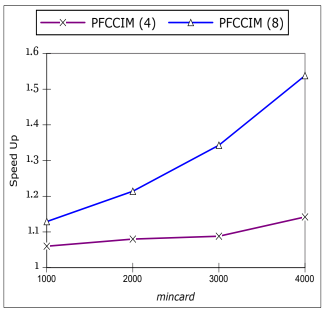
Figure 6.24. Speedup of PFCCIM (4 threads and 8 threads) with respect to PFCCIM (2 threads) for Lung Cancer Dataset



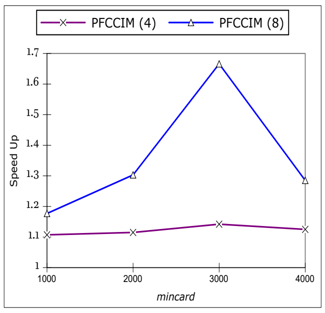
(a) minsup=10



(b) minsup=20

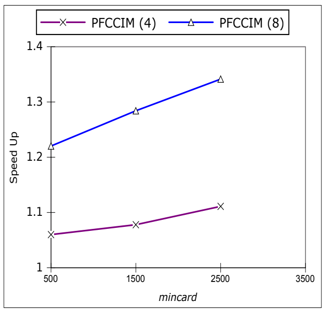


(c) minsup=30

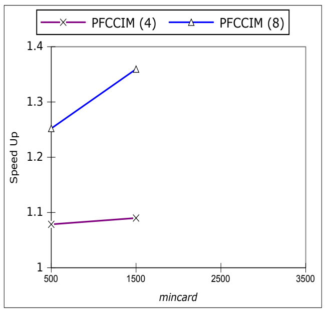


(d) minsup=40

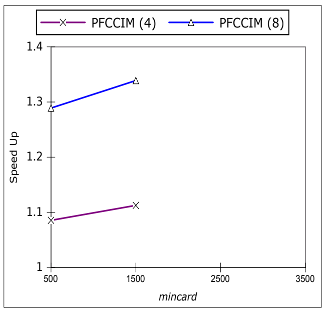
Figure 6.25. Speedup of PFCCIM (4 threads and 8 threads) with respect to PFCCIM (2 threads) for Prostate Cancer Dataset



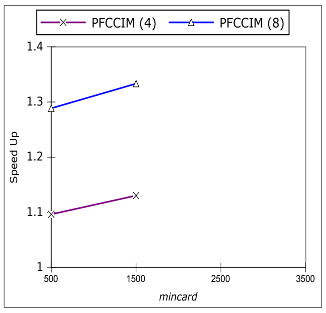
(a) minsup=5



(b) minsup=10

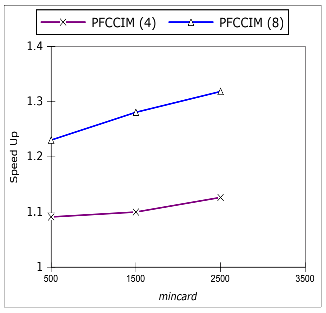


(c) minsup=15

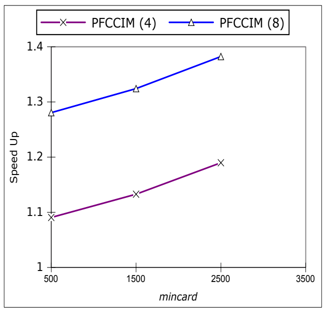


(d) minsup=20

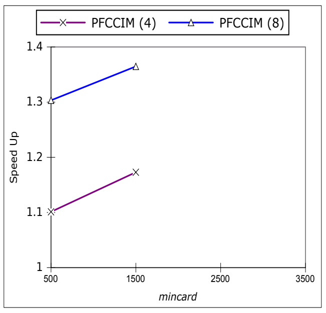
Figure 6.26. Speedup of PFCCIM (4 threads and 8 threads) with respect to PFCCIM (2 threads) for Central Nervous System embryonal tumor Dataset



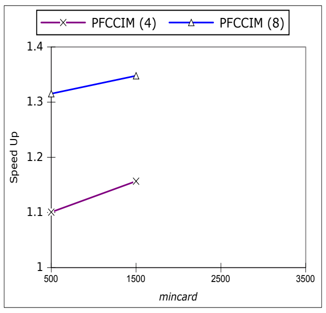
(a) minsup=5



(b) minsup=10



(c) minsup=15



(d) minsup=20

Figure 6.27. Speedup of PFCCIM (4 threads and 8 threads) with respect to PFCCIM (2 threads) for DLBCL (Including Follicular Lymphoma) Dataset

to the PFCCIM algorithm (2 threads) for the prostate cancer dataset, as shown in Figure 6.25, when the (*minsup*, *mincard*) values are set to (10, 2000), (20, 2000), (30, 2000) and (40, 2000) respectively. Similar results have been observed for other experimental datasets for different values of (*minsup*, *mincard*). The solution of mining FCCI from the high dimensional dataset corresponds to the traversal of row enumerated tree. The intrinsic nature of the row enumerated tree, as shown in Figure 4.2, is typically unbalanced. The number of nodes in each branch of row enumerated tree vary. The intrinsic nature of the row enumerated tree will lead to the average speed-up.

The proposed PFCCIM algorithm is not obligatory to gauge the final result when the number of significant rows and features are zero. The number of significant rows and features for the lung cancer dataset is zero when the (*minsup*, *mincard*) reaches (20, 4000), (30, 4000), and (40, 4000). This indicates that there is no speed-up factor when the (*minsup*, *mincard*) reaches (20, 4000), (30, 4000), and (40, 4000), as shown in Figure 6.24b, Figure 6.24c and Figure 6.24d respectively. The number of significant rows and features for central nervous system embryonal tumor dataset is zero when the (*minsup*, *mincard*) reaches (5, 3000), (10, 2500), (15, 2000) and (20, 2000). This indicates that there is no speed-up factor when the (*minsup*, *mincard*) reaches (5, 3000), (10, 2500), (15, 2000), and (20, 2000), as shown in Figure 6.26a, Figure 6.26b, Figure 6.26c and Figure 6.26d respectively. The number of significant rows and features for DLBCL (including the Follicular Lymphoma) dataset is zero when the (*minsup*, *mincard*) reaches (5, 3000), (10, 3000), (15, 2500) and (20, 2500). This indicates that there is no speed-up factor when the (*minsup*, *mincard*) reaches (5, 3000), (10, 3000), (15, 2500), and (20, 2500), as shown in Figure 6.27a, Figure 6.27b, Figure 6.27c, and Figure 6.27d respectively.

6.4.2.3 Statistical Significance Analysis

The experiment results highlight the efficiency of the proposed PFCCIM algorithm over the DisClose algorithm. Figure 6.18, Figure 6.19, Figure 6.20, Figure 6.21, and Figure 6.22 show that PFCCIM (8 threads) outperforms PFCCIM (4 threads) and PFCCIM (2 threads) at different values of *minsup* and *mincard* for ovarian cancer, lung cancer, prostate cancer, central nervous system embryonal tumor, and DLBCL (including the Follicular Lymphoma) datasets respectively. To further analyze the statistical differences among the performance of PFCCIM (2 threads), PFCCIM (4 threads) and PFCCIM (8 threads), the Wilcoxon Singed-Rank statistical significance test has been performed. This test has been selected for the statistical significance analysis as the runtime for PFCCIM (2 threads), PFCCIM (4 threads) and PFCCIM (8 threads) are not normally disturbed and the runtime has been recorded for varying values of *minsup* and *mincard* for ovarian cancer, lung cancer, prostate cancer, central nervous system embryonal tumor, and DLBCL (including the Follicular Lymphoma) datasets, as shown in Figure 6.18, Figure 6.19, Figure 6.20, Figure 6.21, and Figure 6.22 respectively.

Table 6.1 shows the Wilcoxon Signed-Rank test for PFCCIM (2 threads) against PFCCIM (4 threads) and PFCCIM (8 threads) for ovarian cancer, lung cancer, prostate cancer, central nervous system embryonal tumor, and DLBCL (including the Follicular Lymphoma) datasets. Let a null hypothesis indicate that there is no significant difference between the performance of PFCCIM (2 threads) when compared to that of PFCCIM (4 threads) and PFCCIM (8 threads) for a significance level of 5%. When $p \leq 0.05$, the Wilcoxon Signed-Rank test rejects the null hypothesis, indicating that there is a statistically significant differences among samples. When $p > 0.05$, the null hypothesis is retained and it indicates that there is no statistically significant difference among samples. Table 6.1 highlights that the null hypothesis was rejected for ovarian cancer, lung cancer, prostate cancer, central nervous system embryonal tumor, and DLBCL (including the Follicular Lymphoma) datasets. Hence, the difference between the performance of PFCCIM (2 threads), PFCCIM (4 threads) and PFCCIM (8 threads) is statistically significant. From Figure 6.18 - Figure 6.22 and Table 6.1, it is evident that PFCCIM (8 threads) significantly outperforms PFCCIM (4 threads) and PFCCIM (2 threads).

PFCCIM (2 threads), being the least efficient when compared to PFCCIM (4 threads) and PFCCIM (8 threads), is considered for performing statistical significance analysis against the DisClose algorithm. Wilcoxon Singed-Rank Test has been selected for the statistical significance analysis as the runtime for PFCCIM (2 threads) and DisClose algorithm are not normally disturbed and the runtime has been recorded for different values of *minsup* and *mincard* for ovarian cancer, lung cancer, prostate cancer, central nervous system embryonal tumor, and DLBCL (including the Follicular Lymphoma) datasets as shown in Figure 6.13, Figure 6.14, Figure 6.15, Figure 6.16, and Figure 6.17 respectively. Table 6.2 shows the Wilcoxon Signed-Rank Test for PFCCIM (2 threads) against DisClose for ovarian cancer, lung cancer, prostate cancer, central nervous system embryonal tumor, and DLBCL (including the Follicular Lymphoma) datasets. Let a null hypothesis indicates that there is no significant difference between PFCCIM (2 threads) and DisClose for a significance level of 5%. Table 6.2 highlights that the null

Table 6.1. Wilcoxon Signed-Rank Test for PFCCIM (2 threads) against PFCCIM (4 threads) and PFCCIM (8 threads) for Ovarian Cancer, Lung Cancer, Prostate Cancer, Central Nervous System embryonal tumor, and DLBCL (including the Follicular Lymphoma) Dataset.

Dataset	Algorithm	p-value[*]	Null Hypothesis Decision	Significant Difference (if p < 0.05)
Ovarian Cancer	PFCCIM (4 threads)	<0.001	Reject	Yes
	PFCCIM (8 threads)	<0.001	Reject	Yes
Lung Cancer	PFCCIM (4 threads)	<0.002	Reject	Yes
	PFCCIM (8 threads)	<0.002	Reject	Yes
Prostate Cancer	PFCCIM (4 threads)	<0.001	Reject	Yes
	PFCCIM (8 threads)	<0.001	Reject	Yes
Central Nervous System embryonal tumor	PFCCIM (4 threads)	<0.003	Reject	Yes
	PFCCIM (8 threads)	<0.003	Reject	Yes
DLBCL (including the Follicular Lymphoma)	PFCCIM (4 threads)	<0.005	Reject	Yes
	PFCCIM (8 threads)	<0.005	Reject	Yes

[*] p-values are up to three decimal point.

Table 6.2. Wilcoxon Signed-Rank Test for PFCCIM (2 threads) against DisClose for Ovarian Cancer, Lung Cancer, Prostate Cancer, Central Nervous System embryonal tumor, and DLBCL (including the Follicular Lymphoma) Dataset.

Dataset	Algorithm	p-value[*]	Null Hypothesis Decision	Significant Difference (if p < 0.05)
Ovarian Cancer	Disclose	<0.001	Reject	Yes
Lung Cancer	Disclose	<0.001	Reject	Yes
Prostate Cancer	Disclose	<0.001	Reject	Yes
Central Nervous System embryonal tumor	Disclose	<0.002	Reject	Yes
DLBCL (including the Follicular Lymphoma)	Disclose	<0.001	Reject	Yes

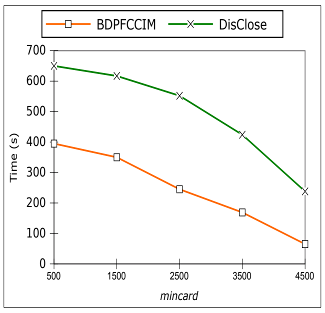
[*] p-values are up to three decimal point.

hypothesis was rejected for ovarian cancer, lung cancer, and prostate cancer dataset. Hence, the difference between the performance of PFCCIM (2 threads) and that of the DisClose algorithm is statistically significant. From Figure 6.13 - Figure 6.17 and Table 6.2 it is evident that PFCCIM (2 threads) significantly outperforms the DisClose algorithm.

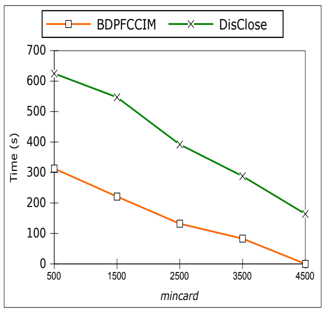
6.4.3 Results of BDPFCCIM Algorithm

This section emphasizes on the efficiency and speed-up of the proposed Balanced Distributed Parallel Frequent Colossal Closed Itemset Mining (BDPFCCIM) algorithm. The proposed BDPFCCIM algorithm has been applied on Mixed Lineage Leukemia (MLL), Central Nervous System embryonal tumor and Diffuse Large B-Cell Lymphoma (DLBCL) including Follicular Lymphoma datasets. The details of the high dimensional datasets have been explained in section 3.3 of chapter 3. The proposed BDPFCCIM algorithm has been compared with the DisClose algorithm. The DisClose algorithm outperforms the other existing FCCI mining algorithms. Hence it is chosen as representative for the experimental evaluation.

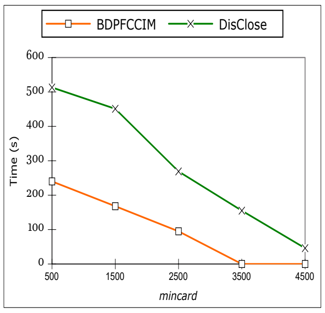
The experiments have been conducted on a cluster consisting of a master node and compute nodes. The master node and compute nodes has an Intel Xeon Phi processor



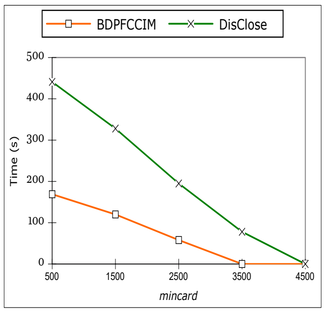
(a) minsup=5



(b) minsup=10

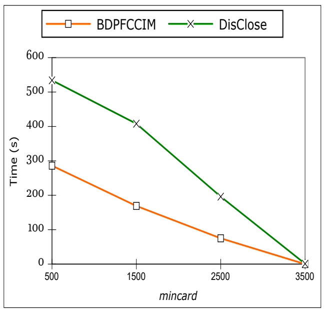


(c) minsup=15

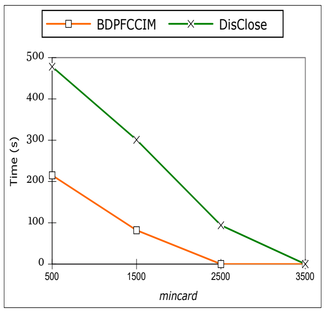


(d) minsup=20

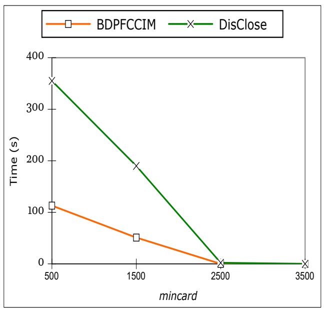
Figure 6.28. Runtime of BDPFCCIM (2 Compute Nodes) (2 threads) and DisClose for MLL Dataset



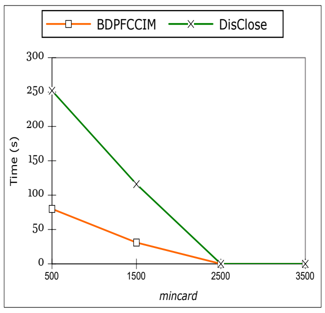
(a) minsup=5



(b) minsup=10

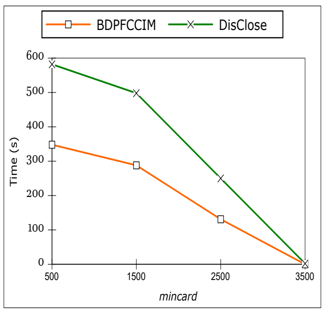


(c) minsup=15

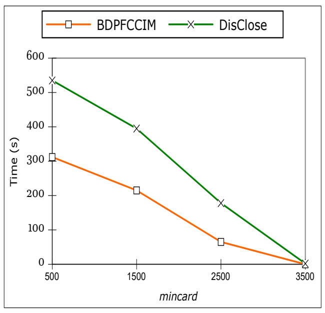


(d) minsup=20

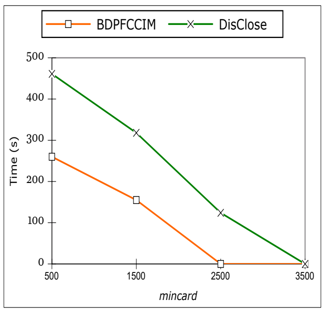
Figure 6.29. Runtime of BDPFCCIM (2 Compute Nodes) (2 threads) and DisClose for Central Nervous System embryonal tumor Dataset



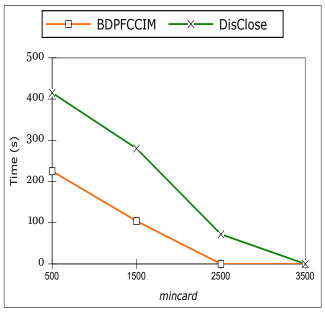
(a) minsup=5



(b) minsup=10



(c) minsup=15



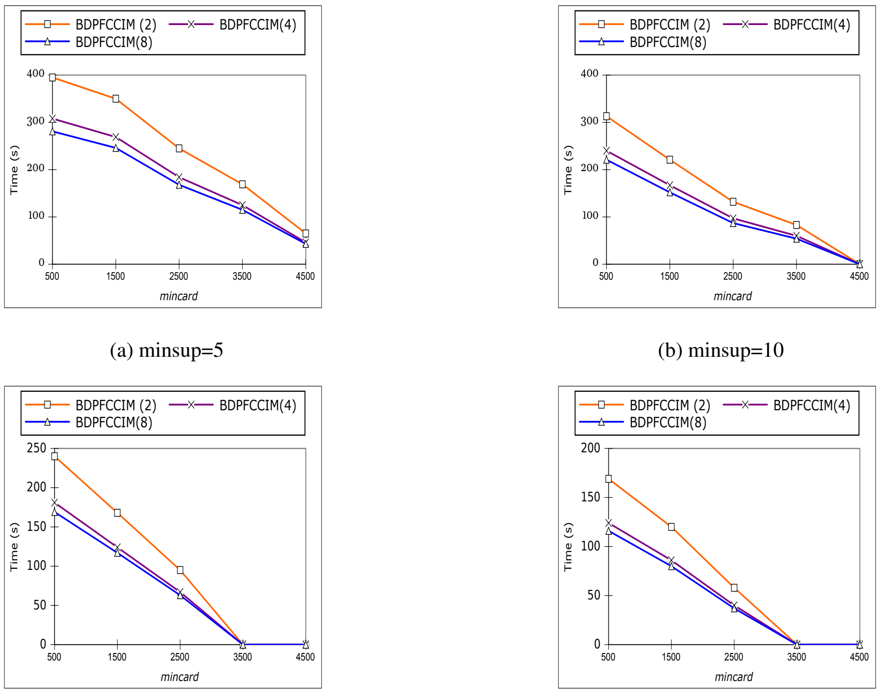
(d) minsup=20

Figure 6.30. Runtime of BDPFCCIM (2 Compute Nodes) (2 threads) and DisClose for DLBCL (including Follicular Lymphoma) Dataset

with 128 GB of RAM. The distributed approach of the proposed BDPFCCIM algorithm has been achieved by using Message Passing Interface (MPI), the standardized message passing library and parallel approach of the proposed BDPFCCIM algorithms has been achieved by using the Open Multi-Processing (OpenMP) application programming interface.

6.4.3.1 Runtime Analysis

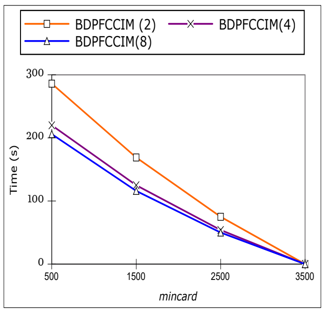
Figure 6.28, Figure 6.29 and Figure 6.30 illustrate the runtime comparison between the DisClose algorithm and the proposed BDPFCCIM algorithm (2 compute nodes, 2 threads) at different values of *minsup* and *mincard* for MLL, central nervous system embryonal tumor and DLBCL (including Follicular Lymphoma) datasets respectively. Figure 6.28, Figure 6.29 and Figure 6.30 illustrate that the proposed BDPFCCIM algorithm (2 compute nodes, 2 threads) outperforms the DisClose algorithm in terms of runtime. Figure 6.31, Figure 6.32, and Figure 6.33 illustrate the runtime comparison between the proposed BDPFCCIM (2 compute nodes, 2 threads), BDPFCCIM (2 compute nodes, 4 threads) and BDPFCCIM (2 compute nodes, 8 threads) at different *minsup* and different *mincard* for MLL, central nervous system embryonal tumor and DLBCL (including Follicular Lymphoma) datasets respectively. Also, it is observed from the



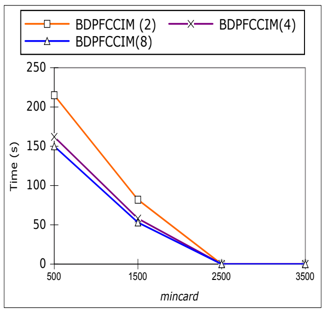
(a) minsup=5 (b) minsup=10

(c) minsup=15 (d) minsup=20

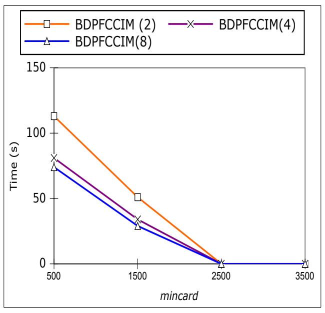
Figure 6.31. Runtime of BDPFCCIM (2 Compute Nodes) (2 threads), BDPFCCIM (2 Compute Nodes) (4 threads) and BDPFCCIM (2 Compute Nodes) (8 threads) for MLL Dataset



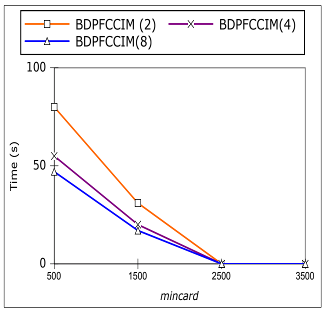
(a) minsup=5



(b) minsup=10

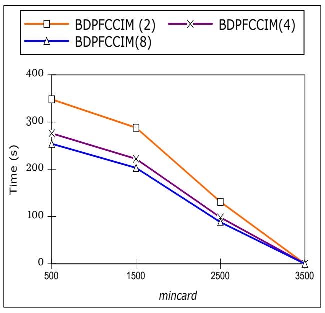


(c) minsup=15

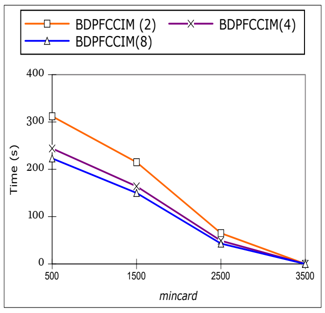


(d) minsup=20

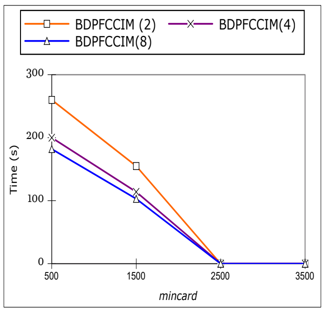
Figure 6.32. Runtime of BDPFCCIM (2 Compute Nodes) (2 threads), BDPFCCIM (2 Compute Nodes) (4 threads) and BDPFCCIM (2 Compute Nodes) (8 threads) for Central Nervous System embryonal tumor Dataset



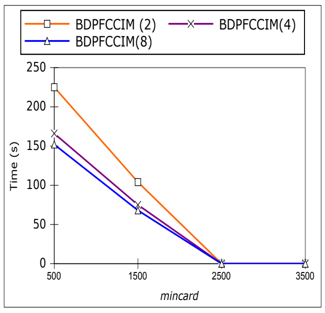
(a) minsup=5



(b) minsup=10



(c) minsup=15

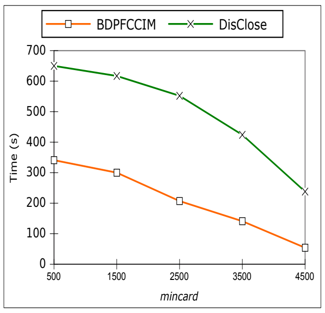


(d) minsup=20

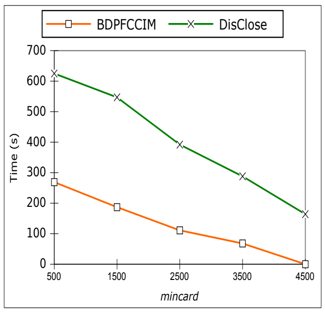
Figure 6.33. Runtime of BDPFCCIM (2 Compute Nodes) (2 threads), BDPFCCIM (2 Compute Nodes) (4 threads) and BDPFCCIM (2 Compute Nodes) (8 threads) for DLBCL (including Follicular Lymphoma) Dataset

Figure 6.31, Figure 6.32, and Figure 6.33 that the BDPFCCIM (2 compute nodes, 8 threads) outperforms BDPFCCIM (2 compute nodes, 4 threads) and BDPFCCIM (2 compute nodes, 2 threads) in terms of runtime.

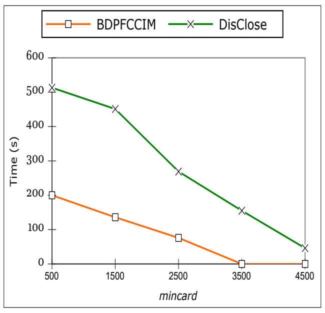
Figure 6.34, Figure 6.35, and Figure 6.36 show the runtime comparison between the DisClose algorithm and the proposed BDPFCCIM algorithm (4 compute nodes, 2 threads) at different *minsup* and different *mincard* for MLL, central nervous system embryonal tumor and DLBCL (including Follicular Lymphoma) datasets respectively. Figure 6.34, Figure 6.35, and Figure 6.36 illustrate that proposed BDPFCCIM algorithm (4 compute nodes, 2 threads) outperforms the DisClose algorithm in terms of runtime. Figure 6.37, Figure 6.38, and Figure 6.39 show the runtime comparison between the proposed BDPFCCIM (4 compute nodes, 2 threads), BDPFCCIM (4 compute nodes, 4 threads) and BDPFCCIM (4 compute nodes, 8 threads) at different *minsup* and different *mincard* for MLL, central nervous system embryonal tumor and DLBCL (including Follicular Lymphoma) datasets respectively. Figure 6.37, Figure 6.38, and Figure 6.39 illustrate that BDPFCCIM (4 compute nodes, 8 threads) outperforms BDPFCCIM (4 compute nodes, 4 threads) and BDPFCCIM (4 compute nodes, 2 threads) in terms of runtime.



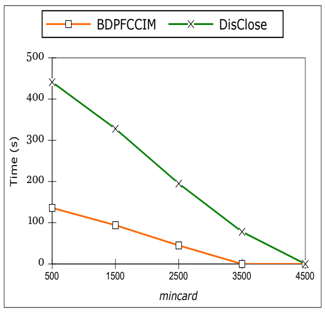
(a) minsup=5



(b) minsup=10

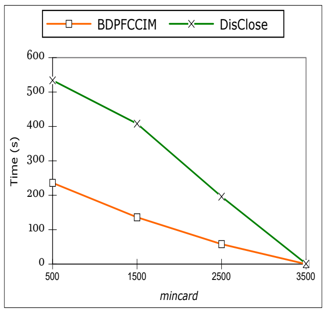


(c) minsup=15

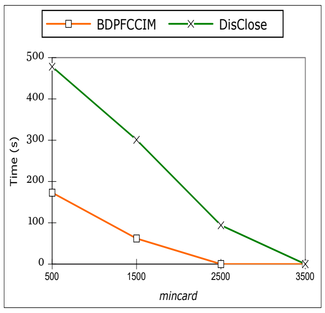


(d) minsup=20

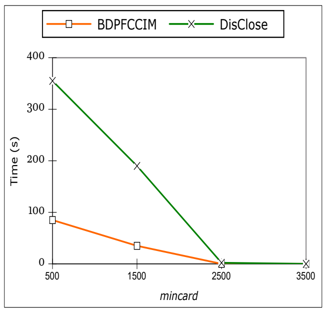
Figure 6.34. Runtime of BDPFCCIM (4 Compute Nodes) (2 threads) and DisClose for MLL Dataset



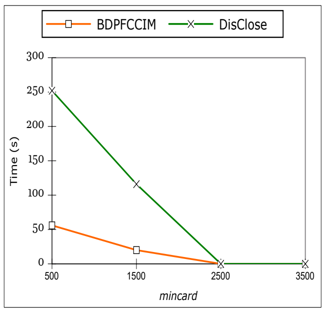
(a) minsup=5



(b) minsup=10

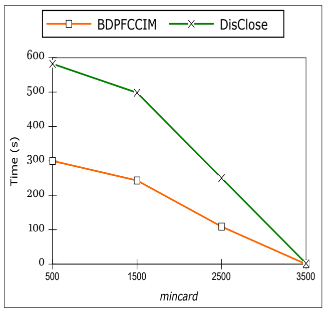


(c) minsup=15

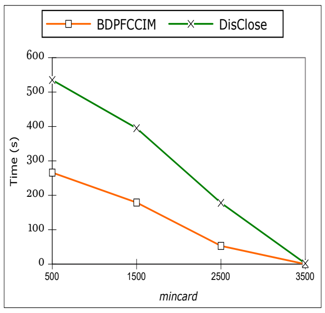


(d) minsup=20

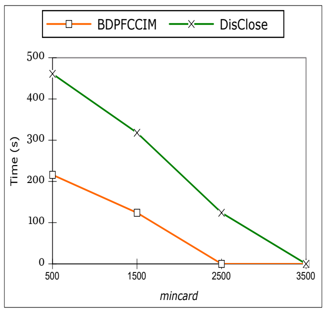
Figure 6.35. Runtime of BDPFCCIM (4 Compute Nodes) (2 threads) and DisClose for Central Nervous System embryonal tumor Dataset



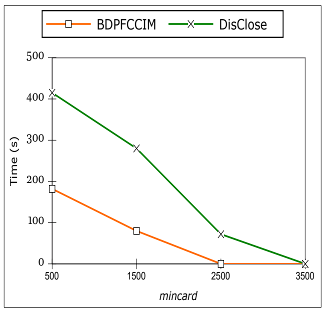
(a) minsup=5



(b) minsup=10

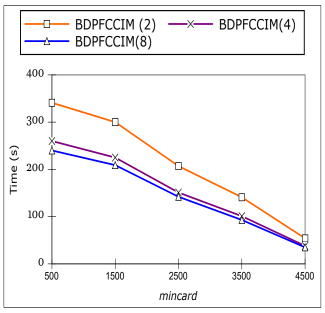


(c) minsup=15

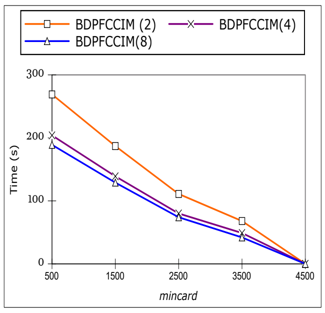


(d) minsup=20

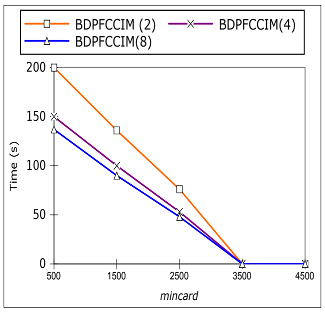
Figure 6.36. Runtime of BDPFCCIM (4 Compute Nodes) (2 threads) and DisClose for DLBCL (including Follicular Lymphoma) Dataset



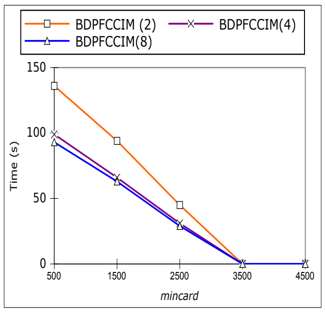
(a) minsup=5



(b) minsup=10

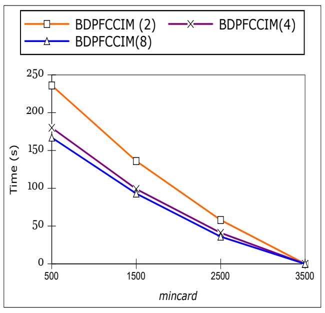


(c) minsup=15

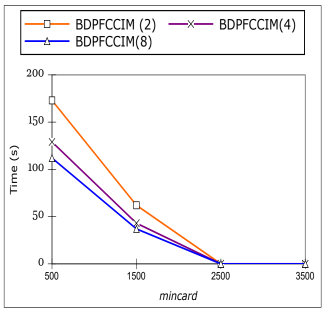


(d) minsup=20

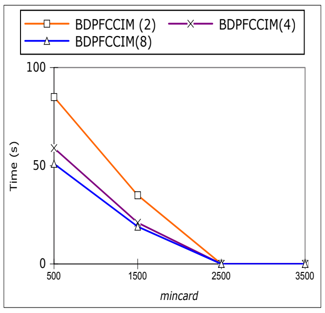
Figure 6.37. Runtime of BDPFCCIM (4 Compute Nodes) (2 threads), BDPFCCIM (4 Compute Nodes) (4 threads) and BDPFCCIM (4 Compute Nodes) (8 threads) for MLL Dataset



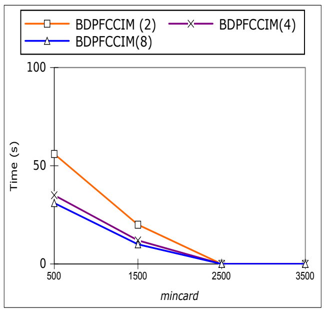
(a) minsup=5



(b) minsup=10

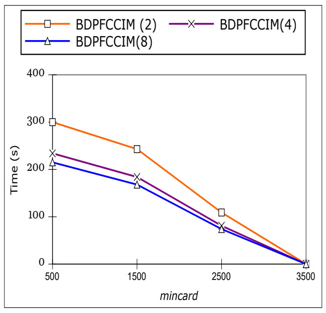


(c) minsup=15

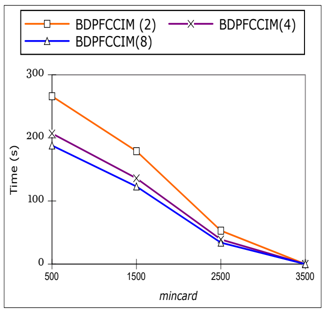


(d) minsup=20

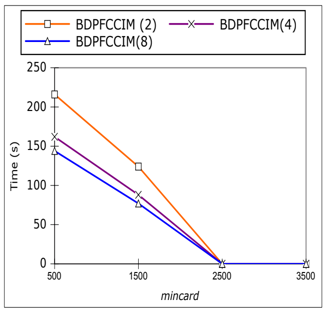
Figure 6.38. Runtime of BDPFCCIM (4 Compute Nodes) (2 threads), BDPFCCIM (4 Compute Nodes) (4 threads) and BDPFCCIM (4 Compute Nodes) (8 threads) for Central Nervous System embryonal tumor Dataset



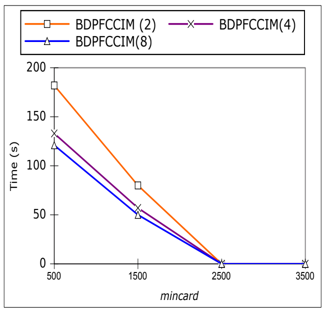
(a) minsup=5



(b) minsup=10

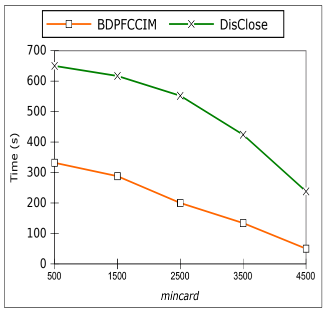


(c) minsup=15

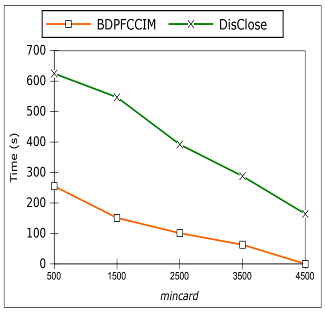


(d) minsup=20

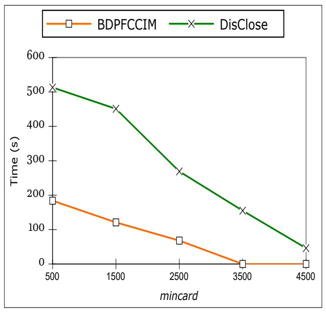
Figure 6.39. Runtime of BDPFCCIM (4 Compute Nodes) (2 threads), BDPFCCIM (4 Compute Nodes) (4 threads) and BDPFCCIM (4 Compute Nodes) (8 threads) for DLBCL (including Follicular Lymphoma) Dataset



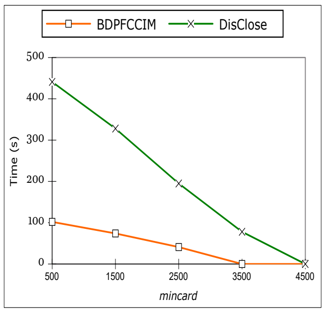
(a) minsup=5



(b) minsup=10

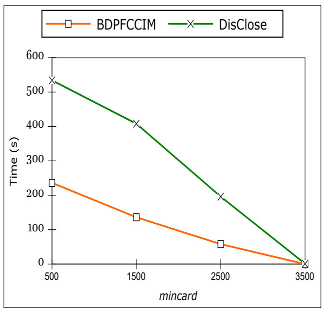


(c) minsup=15

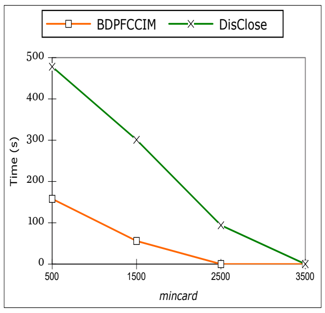


(d) minsup=20

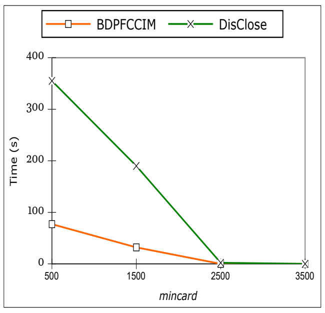
Figure 6.40. Runtime of BDPFCCIM (8 Compute Nodes) (2 threads) and DisClose for MLL Dataset



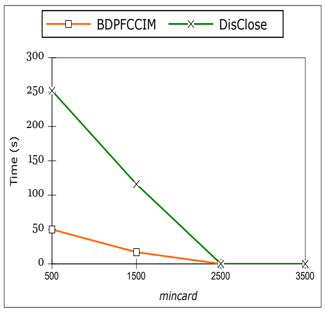
(a) minsup=5



(b) minsup=10

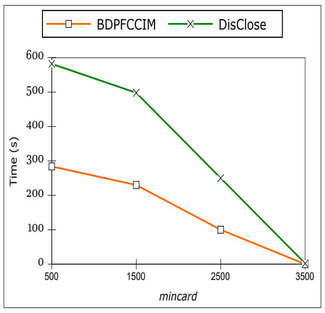


(c) minsup=15

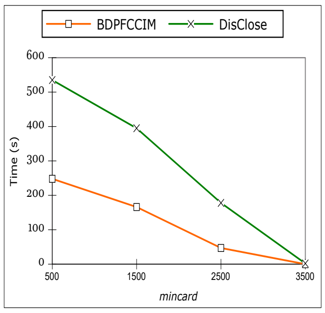


(d) minsup=20

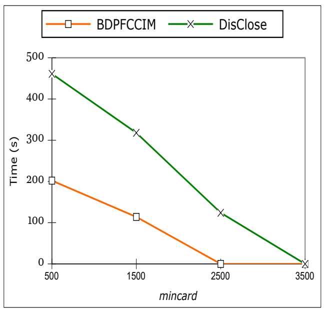
Figure 6.41. Runtime of BDPFCCIM (8 Compute Nodes) (2 threads) and DisClose for Central Nervous System embryonal tumor Dataset



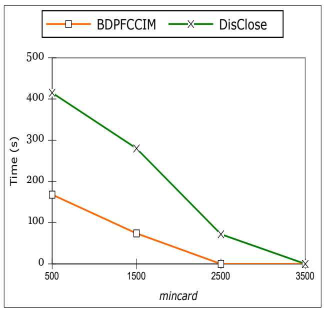
(a) minsup=5



(b) minsup=10

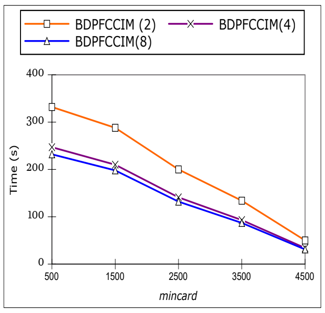


(c) minsup=15

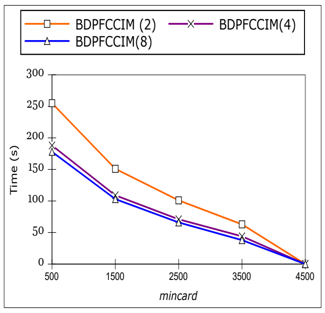


(d) minsup=20

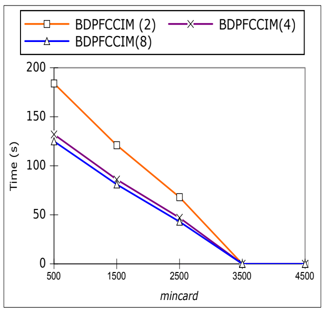
Figure 6.42. Runtime of BDPFCCIM (8 Compute Nodes) (2 threads) and DisClose for DLBCL (including Follicular Lymphoma) Dataset



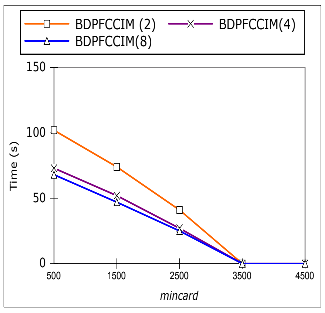
(a) minsup=5



(b) minsup=10

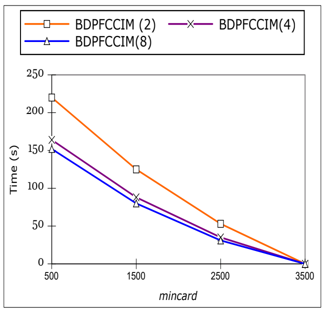


(c) minsup=15

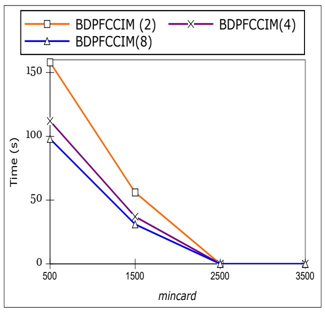


(d) minsup=20

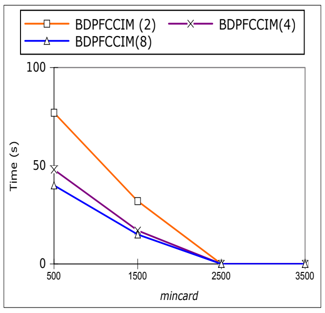
Figure 6.43. Runtime of BDPFCCIM (8 Compute Nodes) (2 threads), BDPFCCIM (8 Compute Nodes) (4 threads) and BDPFCCIM (8 Compute Nodes) (8 threads) for MLL Dataset



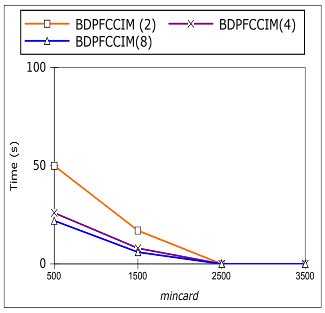
(a) minsup=5



(b) minsup=10

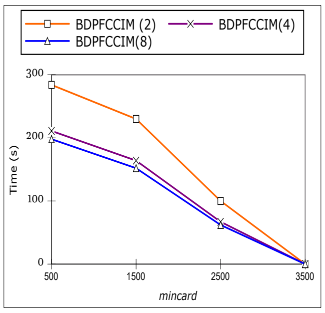


(c) minsup=15

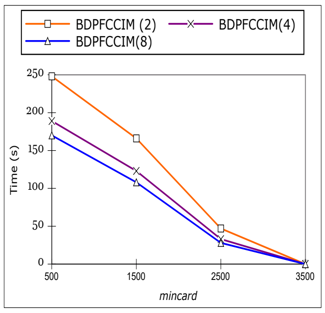


(d) minsup=20

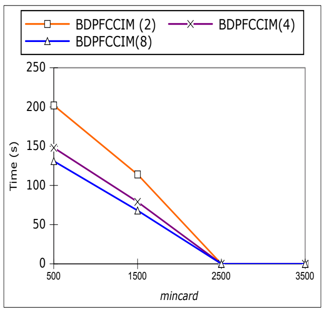
Figure 6.44. Runtime of BDPFCCIM (8 Compute Nodes) (2 threads), BDPFCCIM (8 Compute Nodes) (4 threads) and BDPFCCIM (8 Compute Nodes) (8 threads) for Central Nervous System embryonal tumor Dataset



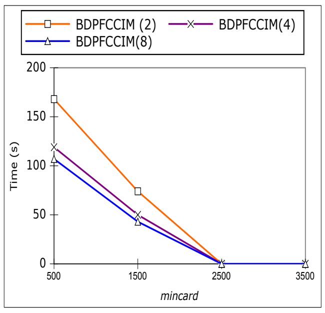
(a) minsup=5



(b) minsup=10



(c) minsup=15



(d) minsup=20

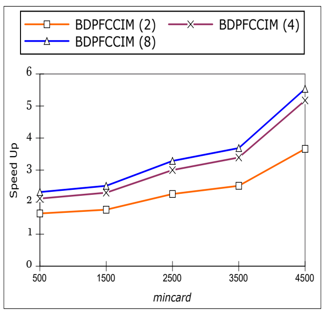
Figure 6.45. Runtime of BDPFCCIM (8 Compute Nodes) (2 threads), BDPFCCIM (8 Compute Nodes) (8 threads) and BDPFCCIM (8 Compute Nodes) (8 threads) for DLBCL (including Follicular Lymphoma) Dataset

Figure 6.40, Figure 6.41 and Figure 6.42 illustrate the runtime comparison between the DisClose algorithm and the proposed BDPFCCIM algorithm (8 compute nodes, 2 threads) at different *minsup* and different *mincard* for MLL, central nervous system embryonal tumor and DLBCL (including Follicular Lymphoma) datasets respectively. Figure 6.40, Figure 6.41 and Figure 6.42 illustrate that the BDPFCCIM algorithm (8 compute nodes, 2 threads) outperforms the DisClose algorithm in terms of runtime. Figure 6.43, Figure 6.44, and Figure 6.45 illustrate the runtime comparison between the proposed BDPFCCIM (8 compute nodes, 2 threads), BDPFCCIM (8 compute nodes, 4 threads) and BDPFCCIM (8 compute nodes, 8 threads) at different *minsup* and different *mincard* for MLL, central nervous system embryonal tumor and DLBCL (including Follicular Lymphoma) datasets respectively. Figure 6.43, Figure 6.44, and Figure 6.45 illustrate that BDPFCCIM (8 compute nodes, 8 threads) outperforms BDPFCCIM (8 compute nodes, 4 threads) and BDPFCCIM (8 compute nodes, 2 threads).

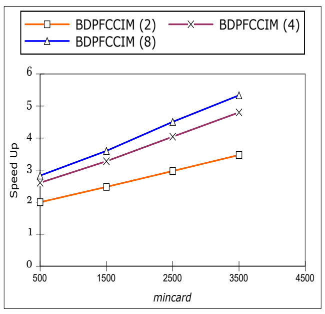
The *x*-axis in Figure 6.28 - Figure 6.45 indicates the different values of *mincard*, and the *y*-axis indicates the runtime. It has been observed from the experimental results that the runtime of the proposed BDPFCCIM algorithm reduces as the *mincard* and *minsup* increases. The experimental results, as shown in Figure 6.28 - Figure 6.45 indicate that the proposed BDPFCCIM algorithm is not obligatory to gauge the final results when the number of significant features and significant rows is zero after applying the proposed EIPP technique. It has been observed from the experimental results that the proposed BDPFCCIM algorithm outperforms the DisClose algorithm in terms of runtime. The RCT based closeness checking method, RCT based pruning strategy, distributed and parallel mining of FCCI helps the proposed BDPFCCIM algorithm to outperform the existing DisClose algorithm. The experimental results as shown in Fig. 6.28 – Fig. 6.45 indicate that for different values *minsup* and *mincard*, the proposed BDPFCCIM with 8 compute nodes outperforms BDPFCCIM with 4 compute nodes and BDPFCCIM with 2 compute nodes.

6.4.3.2 Speed-up Analysis

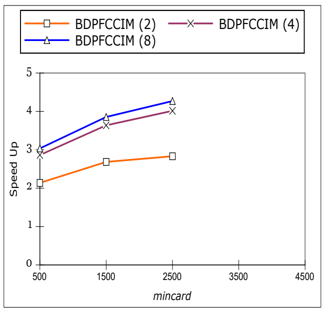
Figure 6.46, Figure 6.47, and Figure 6.48 represent the speed-up of the proposed BDPFCCIM (2 compute nodes, 2 threads), BDPFCCIM (2 compute nodes, 4 threads), and BDPFCCIM (2 compute nodes, 8 threads) with respect to the DisClose algorithm at different values of *minsup* and different *mincard* for MLL, central nervous system embryonal tumor and DLBCL (including Follicular Lymphoma) datasets respectively. The



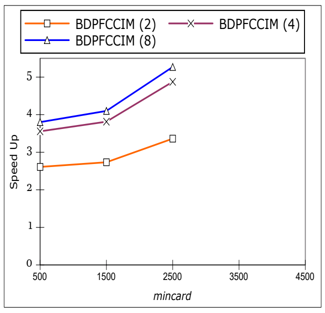
(a) minsup=5



(b) minsup=10

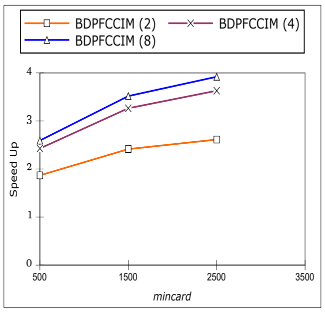


(c) minsup=15

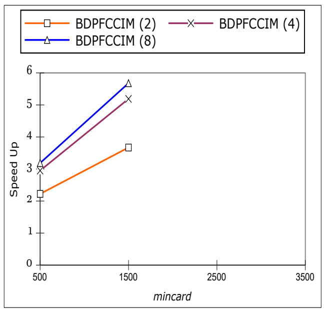


(d) minsup=20

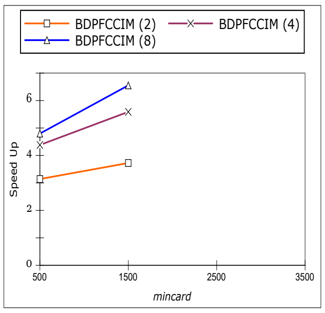
Figure 6.46. Speedup of BDPFCCIM (2 Compute Nodes) (2 threads), BDPFCCIM (2 Compute Nodes) (4 threads) and BDPFCCIM (2 Compute Nodes) (8 threads) with respect to DisClose Algorithm for MLL Dataset



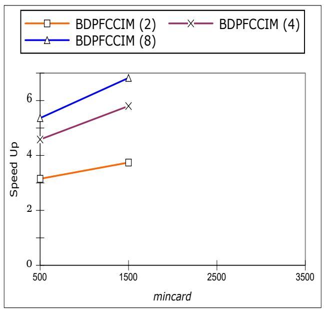
(a) minsup=5



(b) minsup=10

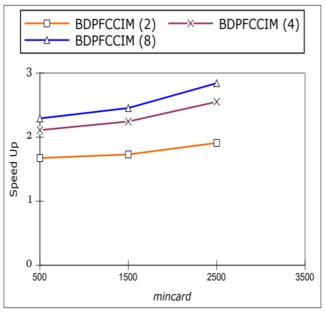


(c) minsup=15

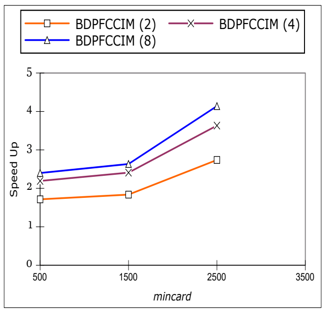


(d) minsup=20

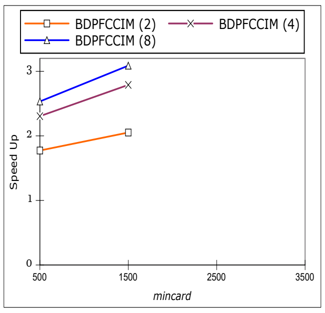
Figure 6.47. Speedup of BDPFCCIM (2 Compute Nodes) (2 threads), BDPFCCIM (2 Compute Nodes) (4 threads) and BDPFCCIM (2 Compute Nodes) (8 threads) with respect to DisClose Algorithm for Central Nervous System embryonal tumor Dataset



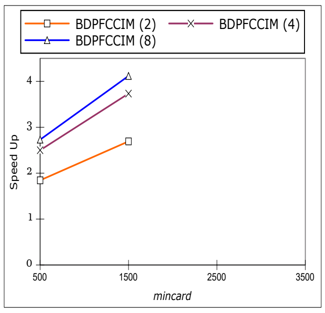
(a) minsup=5



(b) minsup=10

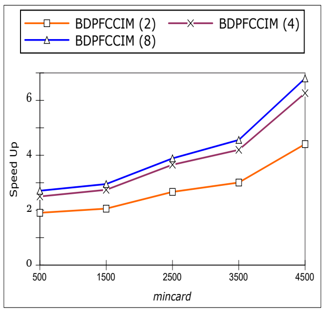


(c) minsup=15

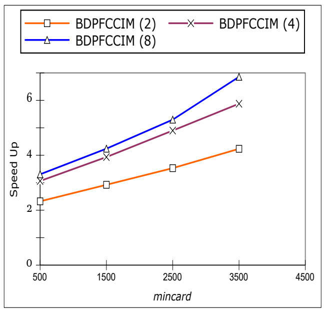


(d) minsup=20

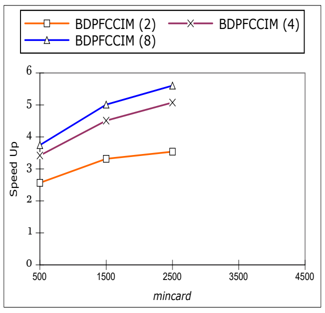
Figure 6.48. Speedup of BDPFCCIM (2 Compute Nodes) (2 threads), BDPFCCIM (2 Compute Nodes) (4 threads) and BDPFCCIM (2 Compute Nodes) (8 threads) with respect to DisClose Algorithm for DLBCL (including Follicular Lymphoma) Dataset



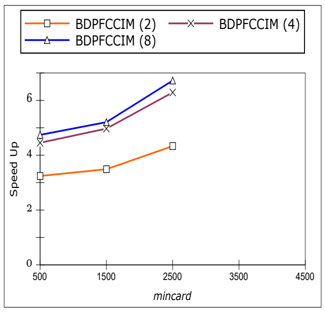
(a) minsup=5



(b) minsup=10

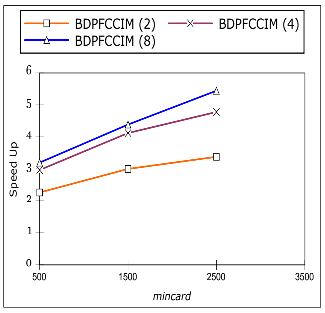


(c) minsup=15

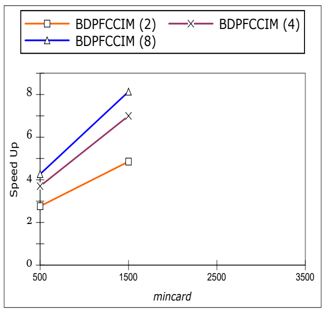


(d) minsup=20

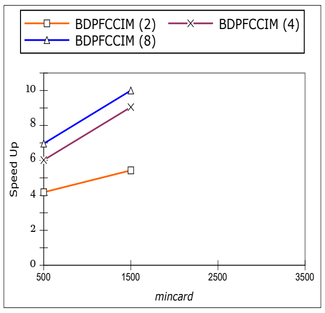
Figure 6.49. Speedup of BDPFCCIM (4 Compute Nodes) (2 threads), BDPFCCIM (4 Compute Nodes) (4 threads) and BDPFCCIM (4 Compute Nodes) (8 threads) with respect to DisClose Algorithm for MLL Dataset



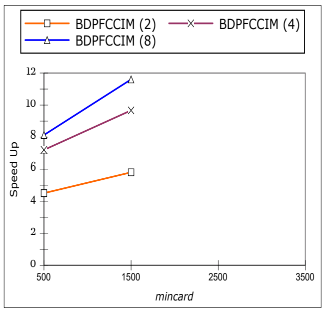
(a) minsup=5



(b) minsup=10

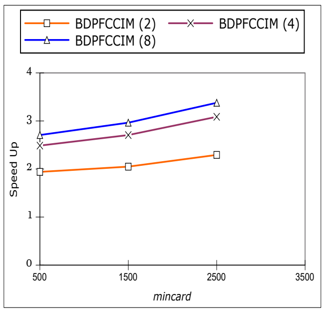


(c) minsup=15

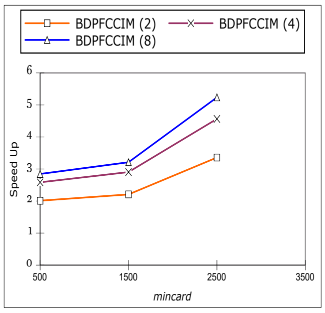


(d) minsup=20

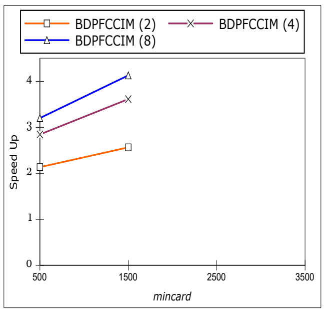
Figure 6.50. Speedup of BDPFCCIM (4 Compute Nodes) (2 threads), BDPFCCIM (4 Compute Nodes) (4 threads) and BDPFCCIM (4 Compute Nodes) (8 threads) with respect to DisClose Algorithm for Central Nervous System embryonal tumor Dataset



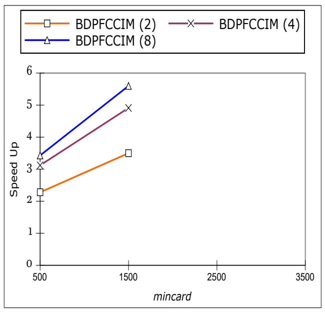
(a) minsup=5



(b) minsup=10

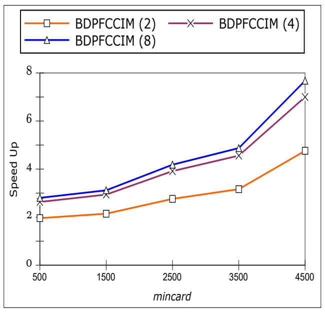


(c) minsup=15

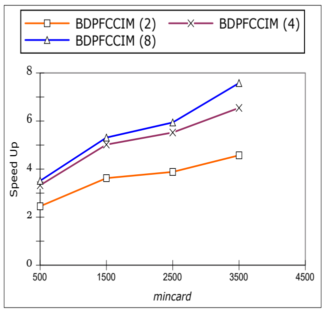


(d) minsup=20

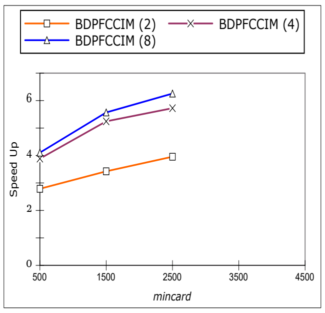
Figure 6.51. Speedup of BDPFCCIM (4 Compute Nodes) (2 threads), BDPFCCIM (4 Compute Nodes) (4 threads) and BDPFCCIM (4 Compute Nodes) (8 threads) with respect to DisClose Algorithm for DLBCL (including Follicular Lymphoma) Dataset



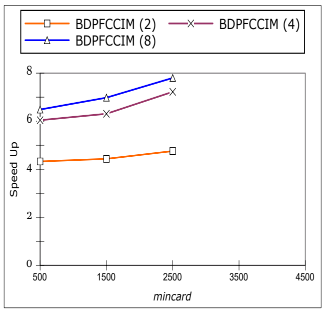
(a) minsup=5



(b) minsup=10

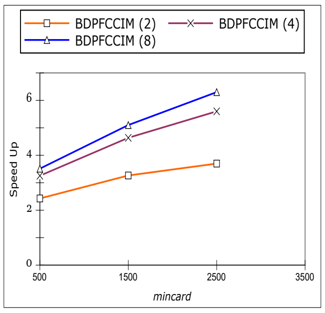


(c) minsup=15

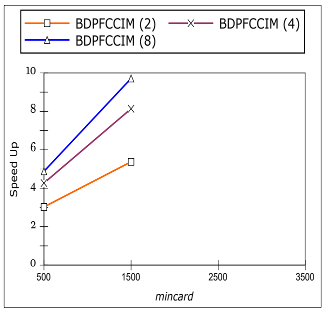


(d) minsup=20

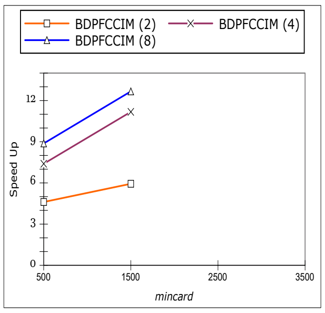
Figure 6.52. Speedup of BDPFCCIM (8 Compute Nodes) (2 threads), BDPFCCIM (8 Compute Nodes) (4 threads) and BDPFCCIM (8 Compute Nodes) (8 threads) with respect to DisClose Algorithm for MLL Dataset



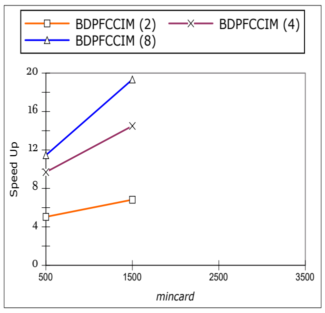
(a) minsup=5



(b) minsup=10

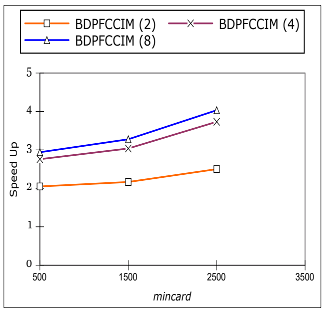


(c) minsup=15

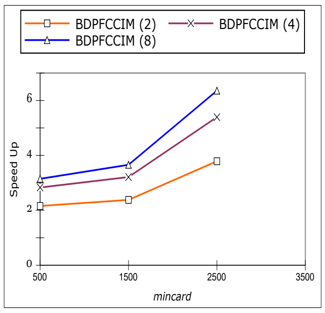


(d) minsup=20

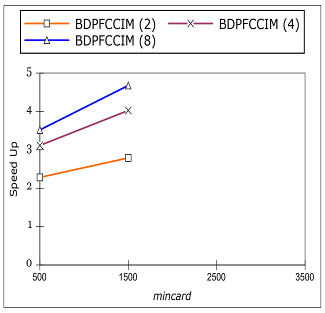
Figure 6.53. Speedup of BDPFCCIM (8 Compute Nodes) (2 threads), BDPFCCIM (8 Compute Nodes) (4 threads) and BDPFCCIM (8 Compute Nodes) (8 threads) with respect to DisClose Algorithm for Central Nervous System embryonal tumor Dataset



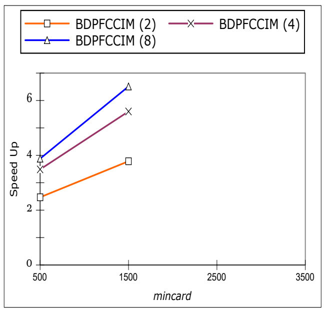
(a) minsup=5



(b) minsup=10



(c) minsup=15



(d) minsup=20

Figure 6.54. Speedup of BDPFCCIM (8 Compute Nodes) (2 threads), BDPFCCIM (8 Compute Nodes) (4 threads) and BDPFCCIM (8 Compute Nodes) (8 threads) with respect to DisClose Algorithm for DLBCL (including Follicular Lymphoma) Dataset

proposed BDPFCCIM algorithm (2 compute nodes, 8 threads) achieves the speed-up of (2.3131, 2.8280, 3.0355, 3.8017) with respect to the DisClose algorithm for the MLL dataset, as shown in Figure 6.46, when the (*minsup*, *mincard*) values are set to (5, 500), (10, 500), (15, 500) and (20, 500) respectively. Similar results have been observed for other experimental datasets for different values of (*minsup*, *mincard*).

Figure 6.49, Figure 6.50, and Figure 6.51 represent the speed-up of the proposed BDPFCCIM (4 compute nodes, 2 threads), BDPFCCIM (4 compute nodes, 4 threads), and BDPFCCIM (4 compute nodes, 8 threads) with respect to the DisClose algorithm at different *minsup* and different *mincard* for MLL, central nervous system embryonal tumor and DLBCL (including Follicular Lymphoma) datasets respectively. The proposed BDPFCCIM algorithm (4 compute nodes, 8 threads) achieves the speed-up of (2.7083, 3.3068, 3.7445, 4.7419) with respect to the DisClose algorithm for the MLL dataset, as shown in Figure 6.49, when the (*minsup*, *mincard*) values are set to (5, 500), (10, 500), (15, 500) and (20, 500) respectively. Similar results have been observed for other experimental datasets for different values of (*minsup*, *mincard*).

Figure 6.52, Figure 6.53, and Figure 6.54 represent the speed-up of the proposed BDPFCCIM (8 compute nodes, 2 threads), BDPFCCIM (8 compute nodes, 4 threads) and BDPFCCIM (8 compute nodes, 8 threads) with respect to the DisClose algorithm at different *minsup* and different *mincard* for MLL, central nervous system embryonal tumor and DLBCL (including Follicular Lymphoma) datasets respectively. The proposed BDPFCCIM algorithm (8 compute nodes, 8 threads) achieves the speed-up of (2.8017, 3.5112, 4.1040, 6.4852) with respect to the DisClose algorithm for the MLL dataset, as shown in Figure 6.52, when the (*minsup*, *mincard*) values are set to (5, 500), (10, 500), (15, 500) and (20, 500) respectively. Similar results have been observed for other experimental datasets for different values of (*minsup*, *mincard*). The x-axis and y-axis in Figure 6.46 - Figure 6.54 indicate the different values of *mincard* and the speed-up respectively.

Figure 6.46, Figure 6.49, and Figure 6.52 shows that the gauging of the final FCCI mining result by the proposed BDPFCCIM is not required for MLL datasets with the *minsup* value set to 15 and *mincard* value set 3500. This indicates that the number of significant rows after applying the proposed EIPP technique is zero and also indicates that there is no speed-up factor. There is no speed-up factor when *minsup* value reaches 10 and *mincard* value reaches 2500 for central nervous system embryonal tumor dataset as shown in Figure 6.47, Figure 6.50, and Figure 6.53. There is no speed-up factor when *minsup* value reaches 15 and *mincard* value reaches 2500 for DLBCL (including Follicular Lymphoma) dataset as shown in Figure 6.48, Figure 6.51, and Figure 6.54. The speed-up factor as shown in Figure 6.46 - Figure 6.54 indicates the proposed BDPFCCIM (8 compute nodes) outperforms BDPFCCIM (4 compute nodes) and BDPFCCIM (2 compute nodes).

6.4.3.3 Statistical Significance Analysis

The experiment results highlight the efficiency of the proposed BDPFCCIM algorithm over the DisClose algorithm. To further analyze the statistical differences among the performance of BDPFCCIM (2 compute nodes, 2 threads), BDPFCCIM (4 compute nodes, 2 threads) and BDPFCCIM (8 compute nodes, 2 threads), the Wilcoxon Singed-Rank statistical significance test has been performed. This test has been selected for the statistical significance analysis as the runtime for BDPFCCIM (2 compute nodes, 2 threads), BDPFCCIM (4 compute nodes, 2 threads) and BDPFCCIM (8 compute nodes, 2 threads) are not normally disturbed and the runtime has been recorded for different values of *minsup* and *mincard* for all the experimental datasets.

Table 6.3 shows the Wilcoxon Signed-Rank test for BDPFCCIM (2 compute nodes, 2 threads) against BDPFCCIM (4 compute nodes, 2 threads) and BDPFCCIM (8 compute nodes, 2 threads) for MLL, central nervous system embryonal tumor, DLBCL (including the Follicular Lymphoma) datasets. Let a null hypothesis indicate that there is no significant difference between the performance of BDPFCCIM (2 compute nodes, 2 threads) when compared to that of BDPFCCIM (4 compute nodes, 2 threads) and BDPFCCIM (8 compute nodes, 2 threads) for a significance level of 5%. When p $\leq$ 0.05, the Wilcoxon Signed-Rank test rejects the null hypothesis, indicating that there is a statistically significant differences among samples. When p $>$ 0.05, the null hypothesis is retained and it indicates that there is no statistically significant difference among samples. Table 6.3 highlights that the null hypothesis was rejected for MLL, central nervous system embryonal tumor, DLBCL (including the Follicular Lymphoma) datasets. Hence, the difference between the performance of BDPFCCIM (2 compute nodes, 2 threads), BDPFCCIM (4 compute nodes, 2 threads) and BDPFCCIM (8 compute nodes, 2 threads) is statistically significant.

From Figure 6.28 - Figure 6.30, Figure 6.34 - Figure 6.36, Figure 6.40 - Figure 6.42 and Table 6.3, it is evident that BDPFCCIM (8 compute nodes, 2 threads) significantly outperforms BDPFCCIM (4 compute nodes, 2 threads) and BDPFCCIM (2 compute

Table 6.3. Wilcoxon Signed-Rank Test for BDPFCCIM (2 compute nodes, 2 threads) against BDPFCCIM (4 compute nodes, 2 threads) and BDPFCCIM (8 compute nodes, 2 threads) for MLL, central nervous system embryonal tumor, DLBCL (including the Follicular Lymphoma) Datasets.

Dataset	Algorithm	p-value*	Null Hypothesis Decision	Significant Difference (if $p < 0.05$)
MLL	BDPFCCIM (4 compute nodes, 2 threads)	<0.0006	Reject	Yes
	BDPFCCIM (8 compute nodes, 2 threads)	<0.006	Reject	Yes
Central Nervous System embryonal tumor	BDPFCCIM (4 compute nodes, 2 threads)	<0.0009	Reject	Yes
	BDPFCCIM (8 compute nodes, 2 threads)	<0.0009	Reject	Yes
DLBCL (including the Follicular Lymphoma)	BDPFCCIM (4 compute nodes, 2 threads)	<0.005	Reject	Yes
	BDPFCCIM (8 compute nodes, 2 threads)	<0.005	Reject	Yes

nodes, 2 threads).

BDPFCCIM (2 compute nodes, 2 threads), being the least efficient when compared to BDPFCCIM (4 compute nodes, 2 threads) and BDPFCCIM (8 compute nodes, 2 threads), is considered for performing statistical significance analysis against the DisClose algorithm. Wilcoxon Singed-Rank Test has been selected for the statistical significance analysis as the runtime for BDPFCCIM (2 compute nodes, 2 threads) and DisClose algorithm are not normally disturbed and the runtime has been recorded for different values of *minsup* and *mincard* for MLL, central nervous system embryonal tumor, DLBCL (including the Follicular Lymphoma) datasets as shown in Figure 6.28, Figure 6.29, and Figure 6.30 respectively. Table 6.4 shows the Wilcoxon Signed-Rank Test for BDPFCCIM (2 compute nodes, 2 threads) against DisClose for for MLL, central nervous system embryonal tumor, DLBCL (including the Follicular Lymphoma)

Table 6.4. Wilcoxon Signed-Rank Test for BDPFCCIM (2 compute nodes, 2 threads) against DisClose for MLL, central nervous system embryonal tumor, DLBCL (including the Follicular Lymphoma) Datasets.

Dataset	Algorithm	p-value*	Null Hypothesis Decision	Significant Difference (if $p < 0.05$)
MLL	Disclose	<0.0001	Reject	Yes
Central Nervous System embryonal tumor	Disclose	<0.0001	Reject	Yes
DLBCL (including the Follicular Lymphoma)	Disclose	<0.0009	Reject	Yes

datasets. Let a null hypothesis indicates that there is no significant difference between BDPFCCIM (2 compute nodes, 2 threads) and DisClose for a significance level of 5%. Table 6.4 highlights that the null hypothesis was rejected for all the experimental datasets. Hence, the difference between the performance of BDPFCCIM (2 compute nodes, 2 threads) and that of the DisClose algorithm is statistically significant. From Figure 6.28, Figure 6.29, Figure 6.30, and Table 6.4 it is evident that BDPFCCIM (2 compute nodes, 2 threads) significantly outperforms the DisClose algorithm.

6.5 Summary

Distributed and parallel computing is a good strategy to overcome the inefficiency of the existing sequential frequent colossal closed itemset mining algorithms. In this chapter, distributed and parallel algorithms have been proposed to efficiently mine FCCI from high dimensional datasets. An efficient Parallel Frequent Colossal Closed Itemset Mining (PFCCIM) algorithm has been proposed to parallelly mine FCCI from the high dimensional dataset. The PFCCIM algorithm has been implemented using the Open Multi-Processing (OpenMP) application programming interface. The experiments have been conducted on various high dimensional datasets. The proposed PFCCIM algorithm (8 threads) achieves the speed-up of (1.223, 1.284, 1.403, 1.435) with respect to the PFCCIM algorithm (2 threads) for the ovarian cancer dataset, when the (*minsup*, *mincard*) values are set to (10, 2000), (20, 2000), (30, 2000) and (40, 2000) respectively. Similar results have been observed for other experimental high dimensional datasets. The proposed PFCCIM algorithm (8 threads) achieves the speed-up of (4.214, 4.298, 6.125, 6.452) with respect to the DisClose algorithm for the ovarian cancer dataset, when the (*minsup*, *mincard*) values are set to (10, 2000), (20, 2000), (30, 2000) and (40, 2000) respectively. Similar results have been observed for other experimental high dimensional datasets. It has been also observed that the results are statistically significant.

The Balanced Distributed Parallel Frequent Colossal Closed Itemset Mining (BDPFCCIM) algorithm has been proposed to handle the unbalanced intrinsic nature of the row enumerated tree. The BDPFCCIM algorithm has been implemented using the Message Passing Interface (MPI) for distributing workloads and Open Multi-Processing (OpenMP) for traversing the row enumerated tree in parallel. The proposed BDPFCCIM algorithm efficiently distributes the branches of row enumerated tree to the compute nodes for traversing and mining FCCI. The branches of the row enumerated tree assigned to the compute nodes are traversed using the parallel bottom-up approach. The proposed BDPFCCIM algorithm (8 compute nodes, 8 threads) achieves the speed-up of (2.8017, 3.5112, 4.1040, 6.4852) with respect to the DisClose algorithm for the MLL dataset, when the (*minsup*, *mincard*) values are set to (5, 500), (10, 500), (15, 500) and (20, 500) respectively. Similar results have been observed for other experimental high dimensional datasets. It has been also observed that the results are statistically significant.

The next chapter concludes our thesis with a summary of the work done and presents some suggestions for further work in this area.

Chapter 7

Conclusions and Future Work

7.1 Conclusions

The high dimensional datasets have attracted interest from researchers to devise a new method to extract significant and important information efficiently. The amount of information that can be extracted from high dimensional datasets is potentially huge, but extraction of information and knowledge from these datasets is a non-trivial task. ARM gives greater importance to the large-sized itemsets called as colossal itemsets. The colossal itemsets are more influential in decision making and are significant in many applications. It is very important to mine colossal itemsets from the high dimensional dataset. The research work in this thesis is directed towards the mining of frequent colossal itemsets and frequent colossal closed itemsets from the high dimensional dataset.

The high dimensional dataset should be preprocessed before the mining of frequent colossal itemsets and FCCI. The complete set of insignificant features and insignificant rows have to be pruned from the high dimensional dataset. The EIP technique based on minimum support threshold and minimum cardinality threshold has been proposed to prune the complete set of insignificant features and insignificant rows from the high dimensional dataset. The proposed EIP technique takes advantage of the reduction in the cardinality of rows due to the pruning of insignificant features and the reduction in the support of features due to the pruning of insignificant rows. The experiments have been conducted on high dimensional datasets for different values of *minsup* and *mincard*. The results show that the proposed EIP technique outperforms the existing preprocessing technique in terms of pruning the complete set of insignificant features and insignificant rows, which further helps in reducing the search space.

The existing frequent colossal itemset mining algorithms mine limited set of frequent colossal itemsets from the high dimensional dataset leading to the generation of an incomplete set of association rules. Frequent colossal itemset mining algorithm has been proposed to achieve better accuracy than existing algorithms in terms of mining number of frequent colossal itemsets from the high dimensional dataset. The proposed frequent colossal itemset mining algorithm outperforms the existing algorithms by achieving better accuracy in mining a greater number of frequent colossal itemsets from the high dimensional dataset. Results show that the proposed algorithm outperforms the BVBUC algorithm by 30% and 29% for the lung cancer test dataset, when the (*minsup*, *mincard*) values are set to (5, 500) and (5, 1000) respectively.

The pruning strategy of the existing frequent colossal closed itemset mining algorithms is inefficient. The drawback has been overcome by proposing the frequent colossal closed itemset mining algorithm enclosed with an efficient pruning strategy. The proposed frequent colossal closed itemset mining algorithm has been enclosed with a pruning strategy to efficiently cut down the row enumerated search space. The pruning strategy takes advantage of the PT to efficiently cut down the row enumerated search space. The experimental results highlight that the proposed frequent colossal closed itemset mining algorithm outperforms the Disclose algorithm in terms of runtime. The proposed frequent colossal closed itemset mining algorithm outperforms the DisClose algorithm by (14, 17, 27, 25) seconds for the lung cancer test dataset, when the (*minsup*, *mincard*) values are set to (5, 1000), (10, 1000), (15, 1000) and (20, 1000) respectively. Similar results have been observed for other experimental high dimensional datasets.

The BitSet Frequent Colossal Closed Itemset Mining (BSFCCIM) algorithm entrenched with the efficient RCT based closeness checking method and pruning strategy has been proposed to efficiently mine FCCI from the high dimensional dataset. The proposed RCT based closeness checking method will not scan through the previously mined FCCI to check the existence and closeness of newly mined frequent colossal itemset. The proposed pruning strategy utilizes the RCT at the row enumerated node to efficiently cut down the search space. The RCT provides the prior information regarding the cardinality of the itemsets to be mined at the descendant nodes without traversing them. It is observed from the experiment results that the proposed BSFCCIM algorithm outperforms the existing DisClose algorithm in terms of runtime. The proposed BSFCCIM algorithm outperforms the DisClose algorithm by (197, 213, 179, 208) seconds for the MLL dataset, when the (*minsup*, *mincard*) values are set to (5, 500), (10, 500), (15, 500) and (20, 500) respectively. Similar results have been observed for other experimental high dimensional datasets.

An efficient Dynamic Switching Frequent Colossal Closed Itemset Mining (DSFCCIM) algorithm has been proposed for mining FCCI from datasets consisting of a large number of rows and a large number of features. The proposed DSFCCIM algorithm efficiently switches between bottom-up row enumerated approach and bottom-up feature enumerated approach based on data characteristics during the mining process. The proposed DSFCCIM algorithm is enclosed with RCT based closeness checking method and pruning strategy; it is also enclosed with IST based closeness checking method and pruning strategy. The proposed DSFCCIM algorithm outperforms the DisClose algorithm by (2516, 2363, 2385, 1610) seconds for the ovarian cancer dataset, when the (*minsup*, *mincard*) values are set to (10, 2000), (20, 2000), (30, 2000) and (40, 2000) respectively. It is observed from the experiment results that the proposed DSFCCIM algorithm outperforms the existing DisClose algorithm in terms of runtime.

Distributed and parallel computing is a good strategy to overcome the inefficiency of the existing sequential frequent colossal closed itemset mining algorithms. Distributed and parallel algorithms have been proposed to efficiently mine FCCI from high dimensional datasets. An efficient Parallel Frequent Colossal Closed Itemset Mining (PFCCIM) algorithm has been proposed to parallelly mine FCCI from the high dimensional dataset. The PFCCIM algorithm has been implemented using the Open Multi-Processing (OpenMP) application programming interface. The experiments have been conducted on various high dimensional datasets. The proposed PFCCIM algorithm (8 threads) achieves the speed-up of (1.223, 1.284, 1.403, 1.435) with respect to the PFCCIM algorithm (2 threads) for the ovarian cancer dataset, when the (*minsup*, *mincard*) values are set to (10, 2000), (20, 2000), (30, 2000) and (40, 2000) respectively. Similar results have been observed for other experimental high dimensional datasets. It has been also observed that the results are statistically significant.

The Balanced Distributed Parallel Frequent Colossal Closed Itemset Mining (BDPFCCIM) algorithm has been proposed to handle the unbalanced intrinsic nature of the row enumerated tree. The BDPFCCIM algorithm has been implemented using the Message Passing Interface (MPI) for distributing workloads and Open Multi-Processing (OpenMP) for traversing the row enumerated tree in parallel. The proposed BDPFCCIM algorithm efficiently distributes the branches of row enumerated tree to the compute nodes for traversing and mining FCCI. The branches of the row enumerated tree assigned to the compute nodes are traversed using the parallel bottom-up approach. The proposed BDPFCCIM algorithm (8 compute nodes, 8 threads) achieves the speed-up of (2.8017, 3.5112, 4.1040, 6.4852) with respect to the DisClose algorithm for the MLL dataset, when the (*minsup*, *mincard*) values are set to (5, 500), (10, 500), (15, 500) and (20, 500) respectively. Similar results have been observed for other experimental high dimensional datasets. It has been also observed that the results are statistically significant.

7.2 Future Work

The work discussed in this thesis has inspired a number of promising directions for future research outlined below.

- Exploring the balanced distributed and parallel dynamic switching algorithm for mining FCCI from datasets that have a large number of features and a large number of rows. The intrinsic nature of the row and feature enumerated tree is typically unbalanced, as the number of nodes in each branch of row and feature enumerated tree vary. It is important to properly distribute the branches of the row and feature enumerated tree among the compute nodes to traverse it and mine the FCCI. The branches of the row and feature enumerated tree assigned to the compute nodes are traversed using the parallel bottom-up approach and the algorithm efficiently switches between bottom-up row enumerated approach and bottom-up feature enumerated approach based on data characteristics during the mining process.

- To discover the strong associations between the mined FCCI and the class attribute, which helps in building the associative classifier for achieving higher classification accuracy. The Class Association Rules (CAR) should be generated in the form of ”IF-THEN” format by identifying the strong associations between the mined FCCI and class attribute. The set of CAR helps is building the classifier.

- Further, the associations between the mined FCCI can be used in gene expression data analysis to uncover the gene networks. A gene network is a set of related genes where expression of one gene may influence the other gene activity. The associations between the mined FCCI can help in understanding the relation between one gene and set of genes. The mined associations can be optimized by the genetic algorithm to analyze the important interactions between genes from gene expression data.

- Designing the associative clustering algorithm for exploring the dependencies between functional genomics datasets. The associative clustering summarizes dependencies between data sets as clusters of similar samples having similar dependencies. Such a method is particularly needed for mining functional genomics data where measurements are available about different aspects of the same set of functioning genes.

References

Aggarwal, C. C., Applications of frequent pattern mining. *In Frequent pattern mining*. Springer, 2014, 443–467.

Agrawal, R., R. Srikant, *et al.*, Fast algorithms for mining association rules. *In Proc. 20th int. conf. very large data bases, VLDB*volume1215. 1994.

Alves, R., D. S. Rodriguez-Baena, and J. S. Aguilar-Ruiz (2009). Gene association analysis: a survey of frequent pattern mining from gene expression data. *Briefings in Bioinformatics*, bbp042.

Amancio, D. R. (2015*a*). A complex network approach to stylometry. *PLoS One*, 10(8), e0136076.

Amancio, D. R. (2015*b*). Probing the topological properties of complex networks modeling short written texts. *PloS one*, 10(2), e0118394.

Ananthanarayana, V., D. Subramanian, and M. N. Murty, Scalable, distributed and dynamic mining of association rules. *In High Performance Computing—HiPC 2000*. Springer, 2000, 559–566.

Aryabarzan, N., B. Minaei-Bidgoli, and M. Teshnehlab (2018). negfin: An efficient algorithm for fast mining frequent itemsets. *Expert Systems with Applications*, 105, 129–143.

Biological-Datasets (). http://datam.i2r.a-star.edu.sg/datasets/krbd/index.html.

Chen, Z., Q. Yan, H. Han, S. Wang, L. Peng, L. Wang, and B. Yang (2017). Machine learning based mobile malware detection using highly imbalanced network traffic. *Information Sciences*.

Chon, K.-W., S.-H. Hwang, and M.-S. Kim (2018). Gminer: A fast gpu-based frequent itemset mining method for large-scale data. *Information Sciences*, 439, 19–38.

Cong, G., K.-L. Tan, A. K. Tung, and F. Pan, Mining frequent closed patterns in microarray data. *In Data Mining, 2004. ICDM'04. Fourth IEEE International Conference on*. IEEE, 2004.

Dabbiru, M. and M. Shashi (2010). An efficient approach to colossal pattern mining. *International Journal Computer Science Network Security*, 10(1), 304–312.

Djenouri, Y. and M. Comuzzi (2017). Combining apriori heuristic and bio-inspired algorithms for solving the frequent itemsets mining problem. *Information Sciences*, 420, 1–15.

Djenouri, Y., D. Djenouri, A. Belhadi, and A. Cano (2018). Exploiting gpu and cluster parallelism in single scan frequent itemset mining. *Information Sciences*.

Dong, J. and M. Han (2007). Bittablefi: An efficient mining frequent itemsets algorithm. *Knowledge-Based Systems*, 20(4), 329–335.

Fayyad, U., G. Piatetsky-Shapiro, and P. Smyth (1996). From data mining to knowledge discovery in databases. *AI magazine*, 17(3), 37.

Fu, H. and M. O. Foghlu, A distributed algorithm of density-based subspace frequent closed itemset mining. *In 2008 10th IEEE International Conference on High Performance Computing and Communications*. IEEE, 2008.

Fumarola, F., P. F. Lanotte, M. Ceci, and D. Malerba (2016). Clofast: closed sequential pattern mining using sparse and vertical id-lists. *Knowledge and Information Systems*, 48(2), 429–463.

Han, J., J. Pei, and Y. Yin, Mining frequent patterns without candidate generation. *In ACM Sigmod Record*volume29. ACM, 2000.

Huang, H., Y. Miao, and J. Shi, Top-down mining of top-k frequent closed patterns from microarray datasets. *In Conference Anthology, IEEE*. IEEE, 2013.

Javed, A. and A. Khokhar (2004). Frequent pattern mining on message passing multiprocessor systems. *Distributed and Parallel Databases*, 16(3), 321–334.

Jiawei Han, M. and J. Pei (2011). Data mining: concepts and techniques: concepts and techniques.

Li, W., J. Han, and J. Pei, Cmar: Accurate and efficient classification based on multiple class-association rules. *In Data Mining, 2001. ICDM 2001, Proceedings IEEE International Conference on*. IEEE, 2001.

Lin, K.-C., I.-E. Liao, T.-P. Chang, and S.-F. Lin (2014). A frequent itemset mining algorithm based on the principle of inclusion–exclusion and transaction mapping. *Information Sciences*, 276, 278–289.

Lin, K. W. and D.-J. Deng (2010). A novel parallel algorithm for frequent pattern mining with privacy preserved in cloud computing environments. *International Journal of Ad Hoc and Ubiquitous Computing*, 6(4), 205–215.

Lin, K. W. and Y.-C. Lo (2013). Efficient algorithms for frequent pattern mining in many-task computing environments. *Knowledge-Based Systems*, 49, 10–21.

Liu, C., Z. Zheng, K.-Y. Cai, and S. Zhang, Distributed frequent closed itemsets mining. *In 2007 Third International IEEE Conference on Signal-Image Technologies and Internet-Based System*. IEEE, 2007.

Liu, G., H. Lu, Y. Xu, and J. X. Yu, Ascending frequency ordered prefix-tree: Efficient mining of frequent patterns. *In Eighth International Conference on Database Systems for Advanced Applications, 2003.(DASFAA 2003). Proceedings.*. IEEE, 2003*a*.

Liu, G., H. Lu, J. X. Yu, W. Wang, and X. Xiao, Afopt: An efficient implementation of pattern growth approach. *In FIMI*. Citeseer, 2003*b*.

Liu, H., J. Han, D. Xin, and Z. Shao, Mining frequent patterns on very high dimensional data: a topdown row enumeration approach. *In Proceeding of the 2006 SIAM international conference on data mining (SDM'06), Bethesda, MD*. SIAM, 2006.

Liu, H., X. Wang, J. He, J. Han, D. Xin, and Z. Shao (2009). Top-down mining of frequent closed patterns from very high dimensional data. *Information Sciences*, 179(7), 899–924.

Lucchese, C., S. Orlando, and R. Perego (2006). Fast and memory efficient mining of frequent closed itemsets. *IEEE Transactions on Knowledge and Data Engineering*, 18(1), 21–36.

Lucchese, C., S. Orlando, and R. Perego, Parallel mining of frequent closed patterns: Harnessing modern computer architectures. *In Data Mining, 2007. ICDM 2007. Seventh IEEE International Conference on*. IEEE, 2007.

Miao, Y., G. Chen, B. Song, and Z. Wang, Tp+ close: Mining frequent closed patterns in gene expression datasets. *In VLDB Workshop on Data Mining and Bioinformatics*. Springer, 2006.

Naulaerts, S., P. Meysman, W. Bittremieux, T. N. Vu, W. V. Berghe, B. Goethals, and K. Laukens (2015). A primer to frequent itemset mining for bioinformatics. *Briefings in bioinformatics*, 16(2), 216–231.

Negrevergne, B., A. Termier, J.-F. Méhaut, and T. Uno, Discovering closed frequent itemsets on multicore: Parallelizing computations and optimizing memory accesses. *In High Performance Computing and Simulation (HPCS), 2010 International Conference on*. IEEE, 2010.

Negrevergne, B., A. Termier, M.-C. Rousset, and J.-F. Méhaut (2014). Para miner: a generic pattern mining algorithm for multi-core architectures. *Data Mining and Knowledge Discovery*, 28(3), 593–633.

Nguyen, T.-L., B. Vo, B. Huynh, V. Snasel, and L. T. Nguyen, Constraint-based method for mining colossal patterns in high dimensional databases. *In International Conference on Information Systems Architecture and Technology*. Springer, 2017*a*.

Nguyen, T.-L., B. Vo, and L. T. Nguyen, A new method for mining colossal patterns. *In Systems, Man, and Cybernetics (SMC), 2016 IEEE International Conference on*. IEEE, 2016.

Nguyen, T.-L., B. Vo, and V. Snasel (2017*b*). Efficient algorithms for mining colossal patterns in high dimensional databases. *Knowledge-Based Systems*, 122, 75–89.

Okubo, Y. and M. Haraguchi, Finding top-n colossal patterns based on clique search with dynamic update of graph. *In Formal Concept Analysis*. Springer, 2012, 244–259.

Pan, F., G. Cong, A. K. Tung, J. Yang, and M. J. Zaki, Carpenter: Finding closed patterns in long biological datasets. *In Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2003.

Pan, F., A. K. Tung, G. Cong, and X. Xu, Cobbler: combining column and row enumeration for closed pattern discovery. *In Scientific and Statistical Database Management, 2004. Proceedings. 16th International Conference on*. IEEE, 2004.

Parsons, L., E. Haque, and H. Liu (2004). Subspace clustering for high dimensional data: a review. *Acm Sigkdd Explorations Newsletter*, 6(1), 90–105.

Pasquier, N., Y. Bastide, R. Taouil, and L. Lakhal, Discovering frequent closed itemsets for association rules. *In Database Theory—ICDT'99*. Springer, 1999, 398–416.

Pei, J., J. Han, R. Mao, *et al.*, Closet: An efficient algorithm for mining frequent closed itemsets. *In ACM SIGMOD workshop on research issues in data mining and knowledge discovery*volume4. 2000.

Prasanna, K. and M. Seetha (2015). Efficient and accurate discovery of colossal pattern sequences from biological datasets: a doubleton pattern mining strategy (dpmine). *Procedia Computer Science*, 54, 412–421.

Qiu, Y., Y.-J. Lan, and Q.-S. Xie, An improved algorithm of mining from fp-tree. *In Machine Learning and Cybernetics, 2004. Proceedings of 2004 International Conference on*volume3. IEEE, 2004.

Rodríguez-González, A. Y., F. Lezama, C. A. Iglesias-Alvarez, J. F. Martínez-Trinidad, J. A. Carrasco-Ochoa, and E. M. de Cote (2018). Closed frequent similar pattern mining: Reducing the number of frequent similar patterns without information loss. *Expert Systems with Applications*, 96, 271–283.

Salah, S., R. Akbarinia, and F. Masseglia (2017). Data placement in massively distributed environments for fast parallel mining of frequent itemsets. *Knowledge and Information Systems*, 53(1), 207–237.

Silva, F. N., D. R. Amancio, M. Bardosova, L. d. F. Costa, and O. N. Oliveira Jr (2016). Using network science and text analytics to produce surveys in a scientific topic. *Journal of Informetrics*, 10(2), 487–502.

Singh, B., R. Singh, N. Kushwaha, and O. Vyas, An efficient approach for discovering closed frequent patterns in high dimensional data sets. *In Advanced Computing, Networking and Informatics-Volume 1*. Springer, 2014, 519–528.

Sohrabi, M. K. and A. A. Barforoush (2012). Efficient colossal pattern mining in high dimensional datasets. *Knowledge-Based Systems*, 33, 41–52.

Sohrabi, M. K. and V. Ghods, Top-down vertical itemset mining. *In Sixth International Conference on Graphic and Image Processing (ICGIP 2014)*. International Society for Optics and Photonics, 2015.

Song, W., B. Yang, and Z. Xu (2008). Index-bittablefi: An improved algorithm for mining frequent itemsets. *Knowledge-Based Systems*, 21(6), 507–513.

Sreedevi, M., G. Vijay Kumar, and L. Reddy, Parallel and distributed approach for incremental closed regular pattern mining. *In IT in Business, Industry and Government (CSIBIG), 2014 Conference on*. IEEE, 2014.

Tanbeer, S. K., C. F. Ahmed, B.-S. Jeong, and Y.-K. Lee (2009). Efficient single-pass frequent pattern mining using a prefix-tree. *Information Sciences*, 179(5), 559–583.

Uno, T., T. Asai, Y. Uchida, and H. Arimura, Lcm: An efficient algorithm for enumerating frequent closed item sets. *In FIMI*volume90. 2003.

Uno, T., M. Kiyomi, and H. Arimura, Lcm ver. 2: Efficient mining algorithms for frequent/closed/maximal itemsets. *In FIMI*volume126. 2004.

Uno, T., M. Kiyomi, and H. Arimura, Lcm ver. 3: collaboration of array, bitmap and prefix tree for frequent itemset mining. *In Proceedings of the 1st international workshop on open source data mining: frequent pattern mining implementations*. ACM, 2005.

Viana, M. P., D. R. Amancio, and L. d. F. Costa (2013). On time-varying collaboration networks. *Journal of Informetrics*, 7(2), 371–378.

Vimieiro, R. and P. Moscato (2014). Disclosed: An efficient depth-first, top-down algorithm for mining disjunctive closed itemsets in high-dimensional data. *Information Sciences*, 280, 171–187.

Vo, B., T.-P. Hong, and B. Le (2012). Dbv-miner: A dynamic bit-vector approach for fast mining frequent closed itemsets. *Expert Systems with Applications*, 39(8), 7196–7206.

Wang, J., J. Han, and J. Pei, Closet+: Searching for the best strategies for mining frequent closed itemsets. *In Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2003.

Wang, S.-Q., Y.-B. Yang, G.-P. Chen, Y. Gao, and Y. Zhang, Mapreduce-based closed frequent itemset mining with efficient redundancy filtering. *In Data Mining Workshops (ICDMW), 2012 IEEE 12th International Conference on*. IEEE, 2012.

Xue, B., D. Lipps, and S. Devineni (2016). Integrated strategy improves the prediction accuracy of mirna in large dataset. *PloS one*, 11(12), e0168392.

Xun, Y., J. Zhang, and X. Qin (2016). Fidoop: Parallel mining of frequent itemsets using mapreduce. *IEEE transactions on Systems, Man, and Cybernetics: systems*, 46(3), 313–325.

Xun, Y., J. Zhang, X. Qin, and X. Zhao (2017). Fidoop-dp: data partitioning in frequent itemset mining on hadoop clusters. *IEEE Transactions on Parallel and Distributed Systems*, 28(1), 101–114.

Yin, X. and J. Han, Cpar: Classification based on predictive association rules. *In Proceedings of the 2003 SIAM International Conference on Data Mining*. SIAM, 2003.

Yoon, Y. and G. G. Lee (2012). Subcellular localization prediction through boosting association rules. *IEEE/ACM transactions on computational biology and bioinformatics*, 9(2), 609–618.

Yu, K.-M. and J. Zhou (2010). Parallel tid-based frequent pattern mining algorithm on a pc cluster and grid computing system. *Expert Systems with Applications*, 37(3), 2486–2494.

Zaki, M. J. and C.-J. Hsiao, Charm: An efficient algorithm for closed itemset mining. *In SDM*volume2. SIAM, 2002.

Zaki, M. J. and C.-J. Hsiao (2005). Efficient algorithms for mining closed itemsets and their lattice structure. *Knowledge and Data Engineering, IEEE Transactions on*, 17(4), 462–478.

Zhang, F., M. Liu, F. Gui, W. Shen, A. Shami, and Y. Ma (2015). A distributed frequent itemset mining algorithm using spark for big data analytics. *Cluster Computing*, 18(4), 1493–1501.

Zhong, N., Y. Li, and S.-T. Wu (2012). Effective pattern discovery for text mining. *IEEE transactions on knowledge and data engineering*, 24(1), 30–44.

Zhou, J. and K.-M. Yu, Balanced tidset-based parallel fp-tree algorithm for the frequent pattern mining on grid system. *In Semantics, Knowledge and Grid, 2008. SKG'08. Fourth International Conference on*. IEEE, 2008*a*.

Zhou, J. and K.-M. Yu, Tidset-based parallel fp-tree algorithm for the frequent pattern mining problem on pc clusters. *In Advances in grid and pervasive computing*. Springer, 2008*b*, 18–28.

Zhu, F., Mining long patterns. *In Frequent Pattern Mining*. Springer, 2014, 83–104.

Zhu, F., X. Yan, J. Han, P. S. Yu, and H. Cheng, Mining colossal frequent patterns by core pattern fusion. *In Data Engineering, 2007. ICDE 2007. IEEE 23rd International Conference on*. IEEE, 2007.

Zulkurnain, N. F., D. J. Haglin, and J. A. Keane, Disclose: discovering colossal closed itemsets via a memory efficient compact row-tree. *In Emerging Trends in Knowledge Discovery and Data Mining*. Springer, 2012, 141–156.

Publications

Journal Papers

1. Manjunath K Vanahalli, Nagamma Patil, "An Efficient Parallel Row Enumerated Algorithm for Mining Frequent Colossal Closed Itemsets from High Dimensional Datasets", Information Sciences, Elsevier, Volume: 496, pp. 343-362. (SCI and Scopus Indexed) (Impact Factor: 5.5)
   DOI: https://doi.org/10.1016/j.ins.2018.08.009

2. Manjunath K Vanahalli, Nagamma Patil, "An Efficient Dynamic Switching Algorithm for Mining Colossal Closed Itemsets from High Dimensional Datasets", Data and Knowledge Engineering, Elsevier. (SCI and Scopus Indexed) (Impact Factor: 1.6) DOI: https://doi.org/10.1016/j.datak.2019.101721

3. Manjunath K Vanahalli, Nagamma Patil, "Distributed Load Balancing Frequent Colossal Closed Itemset Mining Algorithm for High Dimensional Dataset", Journal of Parallel and Distributed Computing, Elsevier. (SCI and Scopus Indexed) (Impact Factor: 1.8) (Revision Submitted) (Under Review)

4. Manjunath K Vanahalli, Nagamma Patil, "An Efficient Colossal Closed Itemset Mining Algorithm for Dataset with High Dimensionality", Journal of King Saud University - Computer and Information Sciences, Elsevier. (Scopus Indexed) (Revision Submitted) (Under Review)

Conference Papers

1. Manjunath K Vanahalli, Nagamma Patil, "Association Analysis of Significant Frequent Colossal Itemsets Mined from High Dimensional Datasets", 3rd IEEE International Conference on Electrical, Computer and Electronics Engineering (UPCON-2016) (pp. 258-263), (IIT-BHU, India) (Scopus Indexed).
   DOI: https://doi.org/10.1109/UPCON.2016.7894662

2. Manjunath K Vanahalli, Nagamma Patil, "Distributed Mining of Significant Frequent Colossal Closed Itemsets from Long Biological Dataset", 18th International Conference on Intelligent Systems Design and Applications (ISDA-2018) (pp. 891-902) (VIT Vellore, India), (Scopus Indexed) (CORE C).
   DOI: https://doi.org/10.1007/978-3-030-16657-1_83

3. Manjunath K Vanahalli, Nagamma Patil, "Colossal Closed Itemset Mining Algorithm for Dataset with High Dimensionality", 11th International Conference on Computing, Communication and Networking, IIT Kharagpur (Scopus Indexed) (Under Review).

Curriculum Vitae

Mr. Manjunath K Vanahalli
Full-Time Research Scholar
Department of Information Technology
National Institute of Technology Karnataka
P.O. Srinivasanagar, Surathkal
Mangalore-575 025

Permanent Address

Manjunath K Vanahalli
Door No. 41, Vinay Colony
Keshwapur, Hubli -580023
Dharwad District, Karnataka, India.
Email: manjunath.k.vanahalli@gmail.com
Mobile: +91-9538156995.

Academic Records

1. M.Tech in Computer Science and Engineering from Manipal University, Manipal, Karnataka, India, 2013.
2. B.E. in Computer Science and Engineering from Basaveshwara Engineering College, Bagalkot, Karnataka, India, 2010.

Research Interests

Data Mining
Bioinformatics
Distributed and Parallel Computing