M.Sc. Engg. Thesis

Design and Implementation of the New Endemism Concept to Determine Special Frequent Patterns

by

Madhusudan Basak
Student No.: 1014052035

Submitted to

Department of Computer Science and Engineering

in partial fulfillment of the requirements for the degree of Master of Science in Computer Science and Engineering

Department of Computer Science and Engineering
Bangladesh University of Engineering and Technology (BUET)
Dhaka 1000

March 28, 2018
Dedicated to my loving parents

Author’s Contact

Madhusudan Basak
Lecturer
Department of Computer Science & Engineering
Bangladesh University of Engineering and Technology (BUET).
Email: madhusudan@cse.buet.ac.bd
The thesis titled “Design and Implementation of the New Endemism Concept to Determine Special Frequent Patterns”, submitted by Madhusudan Basak, Roll No. 1014052035P, Session October 2014, to the Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology, has been accepted as satisfactory in partial fulfillment of the requirements for the degree of Master of Science in Computer Science and Engineering and approved as to its style and contents. Examination held on March 28, 2018.

Board of Examiners

1. Dr. Rifat Shahriyar
   Assistant Professor
   Department of Computer Science and Engineering
   Bangladesh University of Engineering and Technology, Dhaka-1000.

2. Dr. Md. Mostofa Akbar
   Professor and Head of the Department
   Department of Computer Science and Engineering
   Bangladesh University of Engineering and Technology, Dhaka-1000.

3. Dr. Abu Sayed Md. Latiful Hoque
   Professor
   Department of Computer Science and Engineering
   Bangladesh University of Engineering and Technology, Dhaka-1000.

4. Dr. Mohammed Eunus Ali
   Professor
   Department of Computer Science and Engineering
   Bangladesh University of Engineering and Technology, Dhaka-1000.

5. Dr. Swakkhar Shatabda
   Associate Professor
   Department of Computer Science and Engineering
   United International University, Dhaka-1209.
Candidate’s Declaration

This is hereby declared that the work titled “Design and Implementation of the New Endemism Concept to Determine Special Frequent Patterns” is the outcome of research carried out by me under the supervision of Dr. Rifat Shahriyar, in the Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology, Dhaka 1000. It is also declared that this thesis or any part of it has not been submitted elsewhere for the award of any degree or diploma.

Madhusudan Basak
Candidate
Acknowledgment

I express my heart-felt gratitude to my supervisor, Dr. Rifat Shahriyar for his constant supervision of this work. He helped me a lot in every aspect of this work and guided me with proper directions whenever I sought one. His patient hearing of my ideas, critical analysis of my observations and detecting flaws (and amending thereby) in my thinking and writing have made this thesis a success.

I want to thank the members of my thesis committee for their valuable suggestions. I thank Dr. Abu Sayed Md. Latiful Hoque, Dr. Md. Mostofa Akbar, Dr. Mohammed Eunus Ali and specially the external member Dr. Swakkhar Shatabda.

I also want to mention the names of Dr. M. Sohel Rahman and Himel Dev for their valuable support and suggestions.

In this regard, I remain ever grateful to my beloved parents, who always exists as sources of inspiration behind every success of mine I have ever made.
Abstract

In a transaction database or support set, the task of finding out the patterns which occur more frequently than a specified threshold is known as frequent pattern mining. Since its inception in the early 1990s, frequent pattern mining has been extensively studied, and subsequently applied to a wide range of application domains—consumer behavior analysis, web log mining, gene expression profiling to name a few. While there has been substantial research in innovating a wide variety of frequent patterns, the evolution of existing and emergence of new application domains demand to innovate a new variety of patterns that can reveal distinguishing characteristics of the underlying support set. For example, clickstream analysts seek to segment users into meaningful clusters based on their click path, which requires identifying informative click sequences that contribute to user profiling. There are many such examples where analysts seek for patterns that can reveal distinguishing characteristics of the underlying population. As the best of the literature review, there is no recent work to determine these characteristics. But many of these distinguishing characteristics can be identified using a newly proposed concept named endemism. If the constituent elements of a pattern are more likely to be found in combine and less likely to be obtained otherwise, then this co-occurring tendency of these elements will be referred to as endemism and this type of pattern will be called endemic pattern. This thesis introduces this endemism concept to make a pattern level grouping of the records or users, which can provide valuable information about the underlying support set. This work proposes two scoring strategies, Reluctancy Scoring and Affinity Scoring, to evaluate the endemism of the frequent patterns. This thesis also proposes three heuristics, TopK Selection, Optimized Search and Random Selection, as the alternative to the costly Combinatorial Search method for the final grouping of the records. Experiments show that Reluctancy Scoring outperforms Affinity Scoring, and Optimized Search provides the best result among the heuristics with a little sacrifice of time.
Contents

Board of Examiners ii

Candidate’s Declaration iii

Acknowledgment iv

Abstract v

1 Introduction 1
  1.1 Motivation ........................................ 1
  1.2 Problem Statement .................................. 2
  1.3 Objectives and Contributions .......................... 3
  1.4 Solution Overview ................................... 4
  1.5 Applications of Endemism ............................ 5
  1.6 Thesis Organization .................................. 6

2 Background 7
  2.1 Frequent Pattern Mining .............................. 7
    2.1.1 Frequent Itemset Mining ......................... 8
    2.1.2 Sequential Pattern Mining ....................... 9
  2.2 Applications of Frequent Pattern Mining ................. 11
  2.3 Existing Works on Frequent Pattern Mining ................. 13
    2.3.1 Frequent Itemset Mining ....................... 13
2.3.2 Sequential Pattern Mining .............................................. 14
2.3.3 Compressed Representation of the Patterns .................. 15
2.3.4 Constraint Based Pattern Mining ................................. 15
2.4 Summary ........................................................................ 16

3 Problem Formulation ............................................................ 17
3.1 Definitions of the Related Terms ...................................... 17
3.2 Problem Statement .......................................................... 20
3.3 Summary ........................................................................ 21

4 Methodologies ................................................................. 22
4.1 Proposed Technique to Find Endemic Patterns .................. 23
4.2 Frequent Itemset Generation and Scoring ....................... 24
   4.2.1 Frequent Itemset Generation and Scoring Algorithm ........ 24
   4.2.2 Candidate Generation Algorithm ............................... 26
   4.2.3 Endemism Scoring Algorithm: Reluctancy Scoring .......... 26
   4.2.4 Alternative Endemism Scoring Algorithm: Affinity Scoring 28
4.3 Identification of k Number of Groups .............................. 31
   4.3.1 Baseline Algorithm .................................................. 31
   4.3.2 Proposed Heuristic 1 (TopK Selection) ....................... 32
   4.3.3 Proposed Heuristic 2 (Optimized Search) .................... 33
   4.3.4 Proposed Heuristic 3 (Random Selection) ................... 35
   4.3.5 Diversity Scoring Algorithm .................................... 35
   4.3.6 Average Reluctancy Determination ............................ 36
4.4 Summary ........................................................................ 36

5 Experimental Studies .......................................................... 40
5.1 The Dataset .................................................................. 40
   5.1.1 Short Description of the Datasets ............................... 41
   5.1.2 Characteristic Measures of the Datasets .................... 41

vii
5.2 Performance Measures .................................................. 42
5.3 Experimental Setup ..................................................... 44
5.4 Results ................................................................. 46
5.5 Discussion ............................................................. 50

6 Conclusion ................................................................. 61
6.1 Conclusion ............................................................... 61
6.2 Future Work ............................................................ 62
## List of Figures

5.1 Result Showing Comparison of Combinatorial Approach with the Heuristics .......................... 53
5.2 Result Showing Comparison among Different Heuristics on 15000 records of the Instacart Dataset ................................. 54
5.3 Result Showing Comparison among Different Heuristics on 20000 records of the Instacart Dataset ................................. 55
5.4 Result Showing Comparison among Different Heuristics on 50000 records of the Instacart Dataset ................................. 56
5.5 Result Showing Comparison among Different Heuristics on 15000 records of the Online Retail Dataset1 ..................................... 57
5.6 Result Showing Comparison among Different Heuristics on 20000 records of the Online Retail Dataset1 ..................................... 58
5.7 Result Showing Comparison among Different Heuristics on 25000 records of the Online Retail Dataset2 ..................................... 59
5.8 Result Showing Comparison among Different Heuristics on 35000 records of the Online Retail Dataset2 ..................................... 60
List of Tables

2.1 Sample Unordered Transaction Database ..................................... 9
2.2 Sample Unordered Transaction Database ..................................... 10
2.3 Converted Ordered Database ......................................................... 10
3.1 Sample Transaction Database ....................................................... 20
5.1 Different Characteristics of the Datasets ........................................ 43
5.2 Mean Percentage Accuracies and Standard Deviation using Reluctancy Scoring and Affinity Scoring in Different Experiments ........................................ 47
5.3 Mean Diversity Ratios in Different Experiments ................................ 51
5.4 Mean Relative Relevancy Scores in Different Experiments ................. 52
5.5 Mean Execution Time Different Experiments .................................... 52
# List of Algorithms

4.1 Summarized Algorithm .............................................. 23  
4.2 Frequent Itemset Generation and Scoring Algorithm .............. 25  
4.3 Candidate Generation .............................................. 27  
4.4 Endemism Scoring Algorithm : Reluctancy Scoring .............. 29  
4.5 Alternative Endemism Scoring Algorithm: Affinity Scoring ...... 30  
4.6 Combinatorial Algorithm ........................................... 32  
4.7 Heuristic 1 (TopK Selection) Algorithm .......................... 33  
4.8 Heuristic 2 (Optimized Search) Algorithm ....................... 37  
4.9 getBestSharedFactor() Algorithm ................................ 38  
4.10 Heuristic 3 (Random Selection) Algorithm ....................... 39  
4.11 Diversity Scoring Algorithm .................................... 39  
4.12 Average Reluctancy Algorithm .................................. 39
Chapter 1

Introduction

This thesis recognizes the need for a special type of frequent patterns that can reveal distinguishing characteristics of the underlying population. As this need cannot be satisfied by the existing techniques, this thesis introduces a new concept named endemism to identify the patterns mentioned above and apply them in different contexts.

1.1 Motivation

Frequent pattern mining involves identifying patterns such as sets and sequences of events that occur more than a specified number of times in a transaction database. The transactions where a frequent pattern occurs constitutes its support set. Frequent pattern mining comes in many different forms such as itemset, sequential pattern, closed pattern, maximal pattern, and the list goes on. These varieties of patterns are useful for a wide range of applications ranging from e-commerce to healthcare and education.

Some prominent application areas of frequent pattern mining include customer analysis, web log mining, software bug detection, event detection, clustering, classification, spatiotemporal analysis, chemical compound prediction, and biological motif finding. Since the inception of frequent pattern mining, these continually growing application fields have driven and is driving today, many researchers to design more and more efficient techniques to extract different types of frequent patterns from a given set of records.
The most popular school of thought in frequent pattern mining approaches is the Apriori Algorithm [1] (proposed by Agrawal and Srikant in 1994). Then many other algorithms, notably, FP-Growth [2], TreeProjection [3], DHP [4], AprioriAll [5], GSP [6], FreeSpan [7], and PrefixSpan [8] came into play with newer capabilities to address the limitation of the approaches proposed before them. However, all of these algorithms concentrate on some of many aspects in frequent pattern mining, but no single algorithm addresses all aspects of frequent patterns.

Though there are lots of techniques for efficient determination of frequent patterns and efficient representation of them, to the best of the literature review, there is no recent work that aims at identifying a special type of patterns that can reveal unique characteristics of the underlying transactions or support set. For example, in a market basket analysis, one can find that the itemset \( \{A, B, C\} \) are bought frequently and thus a frequent itemset. But it can be the case that item A is bought a large number of times without being bought with \( \{B, C\} \), and same scenario may apply for B and C. On the other hand, if we find that \( \{A, B, C\} \) are most likely to be bought together rather than individually, then it is obviously an interesting pattern and grouping all the transactions that contain this pattern may provide important information about the characteristics of the underlying users. So, this type of grouping is significant for customer buying behavior analysis. Another motivating example can be the segmentation of the users into groups or clusters based on the items they clicked using clickstream database. This process requires identifying informative click sequences that contribute to user profiling. There are many such examples where identification of unique or distinguishing characteristics of the underlying population through the second level or pattern level grouping is required. But none of the existing works are there on pattern level grouping.

1.2 Problem Statement

Our work of this thesis, to the best of our knowledge, is the first one exploring the area of pattern mining where the determination of unique or distinguishing characteristics is done using special frequent patterns. Again, this work is not an alternative to the existing works described above; rather this work uses the efficient frequent pattern mining techniques as an intermediate step to reveal information about the desired characteristics of the records or the user groups.
CHAPTER 1. INTRODUCTION

In this thesis, we build upon the existing works to design a new type of frequent pattern namely endemic pattern and develop efficient algorithms to extract these patterns from datasets. The term Endemism is widely used in Biology. A plant or an animal is endemic to an area means that this plant or animal is mostly found is that area and rare in other areas. For example, Kangaroo is an endemic animal which is found only in Australia continent. This concept can be used here to determine a special type of frequent patterns. A pattern is referred to as endemic pattern if it is most likely to occur in a group rather than individually. That means, if a pattern is purely endemic, then all the elements of the patterns will be found only in the records that this pattern covers and none of the elements will be found outside of those records. Thus, this pattern will be endemic to those records, and this characteristic will uniquely separate the records from other records.

1.3 Objectives and Contributions

The objectives of this thesis are as follows:

1. To introduce a new concept that captures the uniqueness of frequent patterns within their support set, which will be referred to as endemism.

2. To propose a metric to evaluate the endemic quality of the patterns.

3. To propose an efficient approach to identify the endemic patterns from a transaction database.

4. To propose some efficient approaches to identify the desired user groups (groups of records) that have unique characteristics among them, and to evaluate performances of the approaches.

To fulfill the objectives, first of all, this thesis introduces the new endemism concept for the pattern level grouping of the records or users. This pattern level grouping can provide valuable information about the underlying support set. This thesis proposes two scoring strategies, Reluctancy Scoring and Affinity Scoring, to evaluate the endemic quality of the frequent patterns. It provides an efficient framework to determine the endemic patterns from the underlying support set and uses these endemic patterns to identify the desired groups of records or users. This work also uses the ideal Combinatorial Search for the selection of the final groups, and proposes three heuristics, TopK Selection, Optimized
Search and Random Selection, as the alternative to the costly Combinatorial Search method. Experiments show that Reluctancy Scoring outperforms Affinity Scoring, and Optimized Search provides the best result among the heuristics with a little sacrifice of time.

1.4 Solution Overview

In our work, we have considered frequent itemsets as the representative of frequent patterns. A methodology has been developed to achieve the goals. As the itemsets which are not frequent do not have significant grouping capability, so itemsets which are frequent are considered here. The endemic quality of these frequent itemsets are evaluated and finally, a number of itemsets are determined where each itemset forms a group of records. So the overall process consists of three major tasks: determination of frequent itemsets, scoring the endemic property of the itemset and identification of k number of groups. A clever observation is that if any candidate-generation-and-test based approach, e.g. Apriori algorithm [1], is used for the determination of frequent itemsets, then we can merge the first two subtasks into a single subtask: determination and endemism scoring of the frequent itemsets. The endemism scoring determines the co-occurring tendency of the items of the itemset. If it is the case that either all the items co-occur in a record or none of them occurs, then this case is known as pure endemism. But pure endemism is rare in practical cases. So it is important to determine how approximate endemism is shown by the itemsets and it is done through endemism scoring.

Two types of scoring techniques have been proposed in this thesis: Reluctancy scoring and Affinity Scoring. Between these two scoring strategies, Reluctancy scoring performs better than Affinity Scoring as proved by the experiments. The less reluctance score shown by the itemset, the more endemic the itemset in nature. After identifying the n frequent itemsets of length more than one along with their endemism (reluctancy) scores, the next subtask is to determine k number of itemsets that form k number of groups. Three evaluation metrics - Diversity Ratio, Relative Relevancy and Time are used to evaluate the quality of the list of k itemsets determined. The more diversity ratio, more relative relevancy and less execution time are expected.

Different approaches are compared based on these metrics. The ideal strategy is to check all the \( \binom{n}{k} \) combinations and determine the combination that gives the best diversity ratio. This is known
as Combinatorial approach. The Combinatorial approach forms the baseline algorithm but due to its huge time complexity, it is practically infeasible to use. For this reason, three heuristics are proposed: TopK Selection, Optimized Search (OS) and Random Selection. TopK follows a greedy strategy. It sorts the frequent itemsets based on the increasing order of reluctancy score and greedily selects top $k$ itemsets. OS makes a trade-off between Combinatorial and TopK. It considers a number of options including TopK and selects the one that provides the best diversity ratio among these. The Random Selection avoids the sorting task and randomly selects $k$ itemsets from the candidate itemsets. A number of experiments were performed on three different real datasets having diverse characteristics. The experiments showed that Combinatorial provides the best result but it suffers from worst time complexity and sometimes almost impossible to work with. Among the heuristics, OS outperforms the other two heuristics as indicated by the generated diversity ratios. To show the significance of the small difference in the diversity ratios among different approaches, relative relevancy metric was used which also spoke in favor of OS. After OS, it was TopK that showed better diversity ratio and relative relevancy than Random. But if we consider the required execution time, then the ordering was the reverse one. Though Random takes a small amount of time, as it shows very poor performance in both diversity ratio and relative relevance metrics, so it is not recommended to use in practical scenarios.

1.5 Applications of Endemism

Pattern mining is an established field having lots of efficient algorithms for different associated tasks. But the opportunity to find out the unique or distinguishing characteristics of the underlying support set using the pattern level grouping has not been explored yet. We propose a framework that makes a pattern level grouping of the records which can be used to find out many interesting characteristics of the underlying support set. For example, if there is a one-to-one correspondence between the users and records (transactions) in a transaction database, then the groups of records implicitly indicate groups of users. In this case, if the characteristic of some users of a particular group is known, then we can use this knowledge to identify the characteristics of the other unknown users of the group. Moreover, we can exploit the opportunity to predict the buying behavior of the users, make an efficient user based item recommendation system and many more valuable tasks using this type
of information. Thus we are acquiring and applying the insights of the underlying support set just using the grouping of the records. These insights are likely to impact the development of new and emerging tasks in the field of pattern mining.

1.6 Thesis Organization

The full thesis is organized as below.

Chapter 2 provides a brief overview of the two major frequent pattern mining techniques: frequent itemset mining and sequential pattern mining. This chapter also discusses the important applications and a summary of the existing works done in different fields of frequent pattern mining.

Chapter 3 first discusses the definitions of the terms related to the problem and finally provides a formal formulation of the problem of the thesis.

Then Chapter 4 describes the framework of the thesis. The overall strategy and related algorithms are described there.

Chapter 5 lists the information about various experiments performed in this work, their results, and their outcomes. At first, the description of the datasets is provided. Then the performance metrics are introduced. After that, the results of the experiments are shown and evaluated.

Finally, Chapter 6 provides the concluding remarks along with the possible future works of the thesis.
Chapter 2

Background

This chapter provides the background information on different pattern mining techniques and also discusses the key applications of frequent pattern mining. In this chapter, the summaries of the related existing works have also been presented, and the differences between these works and our work have been discussed.

This chapter starts with a brief introduction to the field of frequent pattern mining in Section 2.1. Then, Section 2.2 discusses the key applications of frequent pattern mining. Finally, Section 2.3 provides an overview of the existing related works, draws the limitations of these works to solve our problem and marks the differences between these works with our one.

2.1 Frequent Pattern Mining

Frequent pattern mining is a vastly studied field of data mining. Frequent pattern mining can be described as the extraction of patterns that occur in a significant amount of time in the underlying support set or database. In formal definition, In a database $D$ consisting of records $R_1, R_2, ..., R_n$ where each record contains some elements from the set of elements $E$, frequent pattern mining is the task of finding out patterns $P$ where the elements of $P$ are also from the set of elements $E$ and the number of records containing $P$ in the database is more than a threshold $S$. The threshold used here can be an absolute one or can be relative to the size of the Database. The problem was first discussed in [1] as a part of association rule mining. In that work, frequent pattern mining was the first step
to determine association rules in market basket data. If A and B are two sets of items, then \( A \Rightarrow B \) would be the association rule for a transaction database if

- \( \text{supportCount}(A \cup B) \geq s \)

- \( \frac{\text{supportCount}(A \cup B)}{\text{supportCount}(A)} \geq c \) where \( \text{supportCount}(I) \) refers to the number of transactions that contain the itemset (set of items), I.

After the work of [1], the task of frequent pattern mine gained vast popularity. In that work [1], they considered each transaction as an unordered set of items and extracted the itemsets that are frequent in the database of transactions. This process is called \textit{Frequent Itemset Mining}. A related problem arose if we consider ordering among the items or elements in the transactions. This is known as \textit{Sequential Pattern Mining} or \textit{Frequent Subsequence Mining}. Mining transactions having temporal ordering among the elements can be an example of sequential pattern mining. Again, if the records of the support set can be represented as graphs, then finding subgraphs which fulfill the frequency (support) threshold constraint is referred to as \textit{Frequent Subgraph Mining}. There are some other types of mining which differ on the representation of the underlying dataset. All these different mining subdomains are commonly known as \textit{Frequent Pattern Mining}.

In the next subsection, we shall discuss about two major categories of frequent pattern mining - frequent itemset mining and sequential pattern mining.

### 2.1.1 Frequent Itemset Mining

In case of frequent itemset mining, each transaction or record of the database consists of an unordered set of items or elements. For example, consider the sample transaction database of Table 2.1. In this table, the first column contains the transaction ids and the second column contains the list of items bought together in that transaction. Now, the task of frequent itemset mining is to identify the itemsets (sets of items) that occur more than a predefined threshold or support threshold. Suppose, the support threshold is 2. This is an example of absolute support count while an alternative strategy can be using relative support threshold, i.e., the support threshold that is defined as the certain fraction of the number of transactions. However, the frequent itemsets fulfilling the support threshold in the running
example are \{A\}, \{B\}, \{C\}, \{D\}, \{X\}, \{A, B\} and \{C, D\}. As there is no ordering among the items in the transactions, so the transaction having transaction id 2 is considered as the one containing the itemset \{A, B\}. Among the obtained itemsets, \{A\}, \{B\}, \{C\}, \{D\} and \{X\} are frequent itemset of length 1. \{A, B\} and \{C, D\} are frequent itemset of length 2 and there is no frequent itemset of length 3 or more for this example.

<table>
<thead>
<tr>
<th>Transaction ID</th>
<th>Item List</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>{A, B, C, D}</td>
</tr>
<tr>
<td>2</td>
<td>{B, A, M, N, X}</td>
</tr>
<tr>
<td>3</td>
<td>{C, E, F}</td>
</tr>
<tr>
<td>4</td>
<td>{X, Y, C, D}</td>
</tr>
<tr>
<td>5</td>
<td>{P, Q, R, S}</td>
</tr>
</tbody>
</table>

Table 2.1: Sample Unordered Transaction Database

2.1.2 Sequential Pattern Mining

Sequential Pattern Mining is also known as Frequent Subsequence Mining. For sequential pattern mining, the elements of the records of the database contain strict ordering among them. In this case, each element can be a single item or a set of items. For example, consider the transaction database of Table 2.2.

Now, if we group the transactions based on customers and sort the transactions of a customer based on the increasing order of time, then we shall get the converted ordered database as depicted in Table 2.3 where the items inside the \(<\) and \(>\) are bought together, and so there is no ordering among the items bought together.
<table>
<thead>
<tr>
<th>Customer ID</th>
<th>Transaction ID</th>
<th>TimeStamp</th>
<th>Item List</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>T1</td>
<td>{A, B}</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>T2</td>
<td>{B}</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>T3</td>
<td>{C, D}</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>T4</td>
<td>{C}</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>T5</td>
<td>{A}</td>
</tr>
<tr>
<td>4</td>
<td>6</td>
<td>T6</td>
<td>{X, Y}</td>
</tr>
<tr>
<td>5</td>
<td>7</td>
<td>T7</td>
<td>{P, Q, R}</td>
</tr>
<tr>
<td>2</td>
<td>8</td>
<td>T8</td>
<td>{M, N}</td>
</tr>
<tr>
<td>3</td>
<td>9</td>
<td>T9</td>
<td>{E, F}</td>
</tr>
<tr>
<td>5</td>
<td>10</td>
<td>T10</td>
<td>{S}</td>
</tr>
<tr>
<td>4</td>
<td>11</td>
<td>T11</td>
<td>{C, D}</td>
</tr>
<tr>
<td>2</td>
<td>12</td>
<td>T12</td>
<td>{X}</td>
</tr>
</tbody>
</table>

Table 2.2: Sample Unordered Transaction Database

<table>
<thead>
<tr>
<th>Customer ID</th>
<th>Item List</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>&lt;A, B&gt; &lt;C, D&gt;</td>
</tr>
<tr>
<td>2</td>
<td>&lt;B&gt; &lt;A&gt; &lt;M, N&gt; &lt;X&gt;</td>
</tr>
<tr>
<td>3</td>
<td>&lt;C&gt; &lt;E, F&gt;</td>
</tr>
<tr>
<td>4</td>
<td>&lt;X, Y&gt; &lt;C, D&gt;</td>
</tr>
<tr>
<td>5</td>
<td>&lt;P, Q, R&gt; &lt;S&gt;</td>
</tr>
</tbody>
</table>

Table 2.3: Converted Ordered Database

For this sorted database, if we consider the absolute support threshold as 2, then A, B, C, D and X are frequent subsequence of length 1, only CD is the frequent subsequence of length 2 and there is no frequent subsequence of length 3 or more. A careful observation can find that as there is a strict ordering among the elements of records, and so the 2nd customer record does not support the
subsequence AB but supports the subsequence BA.

2.2 Applications of Frequent Pattern Mining

Frequent pattern mining is considered as one of the most important data mining technique having a huge number of applications in real life. In fact, frequent pattern mining is mostly used as an efficient intermediate tool that provides interesting pattern-based insights for other data mining applications. Some of the application domains and related problem areas where frequent pattern mining is the widely accepted according to [9] are described in brief below.

1. **Customer Analysis:** The first and most motivating example of frequent pattern mining is customer analysis [10]. In market basket analysis, the buying behavior of the customers can be identified mainly in two ways: firstly, by frequent itemset mining considering the baskets of items bought together, and secondly, by sequential pattern mining [5] on the sequences of items bought in succession and ordered by time. In all cases, the target is to identify the customer buying pattern or buying behavior which can be very much important for the market owners through recommendations and other vital decision making purposes.

2. **Classification, Clustering and Outlier Analysis:** Frequent pattern mining in a facilitator to other major mining problems such clustering [11], classification [12] [13] and outlier analysis [14]. Frequent pattern mining techniques are used for subspace clustering where each pattern corresponds to a rectangular region in the subspace of the underlying data. In case of classification, the target is to identify the discriminative patterns and use them to obtain rules where an obtained pattern resides on the left side and class variable resides on the right side of the rule. Frequent pattern mining techniques are also very popular in determining outliers in the data.

3. **Indexing and Retrieval:** Frequent pattern mining techniques can be used for indexing and retrieval purposes in case of market basket analysis. Since indexing algorithms typically require the summarized representation of the data, frequent pattern mining is an obvious choice for this purpose [15] [16] [17].
4. **Web Mining**: Sequential pattern mining techniques are heavily used in the field of web mining. Different methods [18] [19] [20] [21] [22] [23] [24] [25] [26] use different techniques of pattern mining to find out the important traversal patterns from the web log. These traversal patterns are the indicator of browsing behaviors which can play a significant role in the improvement of web usage through effective website design, recommendation, outlier analysis using anomaly detection etc. Web linkage mining [27] [28] is another field where frequent pattern mining techniques are commonly used.

5. **Software Bug Analysis**: Software programs can be thought as structured entities which form software behavior graphs. Different frequent pattern mining techniques are deployed on these software behavior graphs to determine important logical errors [29] [30]. Bugs in the execution of software in sensor networks can also be identified using pattern mining techniques [31].

6. **Chemical and Biological Applications**: Several chemical compounds consisting of atoms and bonds can be represented as graphs. So frequent pattern mining in graphs, i.e., frequent subgraph mining can be used to reveal valuable informative structures and other important properties [32] [33] [34] using the graphs. On the other hand, some biological data such as biological molecules, microarrays, protein interaction networks etc. can be represented either as sequences or as graphs. So sequential pattern mining techniques and frequent subgraph mining techniques have been proved fruitful in such cases [35] [36] [37] [38] [39] [40] [41] [42]. Examples of important biological analysis are phylogenetics analysis, glycines analysis, RNA analysis etc.

7. **Spatial and Spatiotemporal Applications**: Frequent pattern mining techniques have been successfully applied for several spatial and spatiotemporal applications. Determining clusters [43] [44] [45] [46] in different spatiotemporal data, identifying frequent patterns from trajectory data and finding out rare classes through classification [47] using trajectory data are some interesting applications of frequent pattern mining. Another worth mentionable application is the discovery of spatial association rules between spatial and non-spatial attribute values of the spatial data [48] [49] [50] [51].
8. **Other Temporal Applications**: Different types of frequent pattern mining algorithms are used as subroutines in several temporal applications such as periodic pattern mining, event detection etc. [52] [53] [54] [55] [56].

### 2.3 Existing Works on Frequent Pattern Mining

After the legendary work of Apriori [1], various research works were performed in different fields of frequent pattern mining, such as frequent itemset mining, sequential pattern mining, compressed representation of the obtained patterns, constraint based pattern mining etc.

#### 2.3.1 Frequent Itemset Mining

In the field of frequent itemset mining, the first work was Apriori [1]. Apriori was a join-based algorithm generating (k+1)-candidate patterns from frequent k-patterns. Actually, all the algorithms of frequent itemset mining implicitly or explicitly traverse the enumeration tree of frequent patterns. Apriori used a breadth-first search or level-wise traversal in the tree. The original algorithm was computationally intensive and possessed the risk of facing the problem of pattern explosion. So two optimizations, AprioriTid and AprioriHybrid, were provided in the same work. Other optimizations over Apriori were introduced in [57] and [58]. Later the DHP algorithm [4], another join-based technique, proposed two more optimizations over Apriori algorithm. The first optimization used there was to prune unpromising candidate itemsets in each iteration and the second one was to delete irrelevant items from the transaction thus reducing the size of transactions. Among the other improved works, [3] used a clever technique for 2-itemset counting, [59] used a set-theoretic approach to calculate the lower bound of supports to skip the counting process of an itemset and [60] used hypercube decomposition technique to find the support count of multiple frequent patterns at a time.

The observation that frequent itemset generation is actually the construction of enumeration tree of frequent patterns led the discovery of more efficient tree-based algorithms. [61] and [3] used clever projection of the transactions during the traversal of the tree. The depth-first variation of projection strategies can be recursively implemented by extending either the prefix or the suffix of the frequent items. [61] and [3] are the examples of such prefix-based pattern growth. On the other
hand, FP-Growth uses suffix-based pattern exploration with the help of FP-tree. [62] [63] [64] [65] [66] [67] [68] [69] [70] [71] introduced efficient construction techniques of FP-tree for suffix-based pattern growth.

Apart from the techniques discussed above, several works using some latest optimizations have been incorporated in this field of frequent itemset mining, e.g., [72] [73] [74] [75] [76] etc.

**Limitations:** All these works have concentrated on identifying frequent itemsets from the underlying support set or transactions which is not the goal of this thesis. The target of this thesis is to find out the itemsets that shows the co-occurring tendency and thus group of transactions so that ultimately the unique characteristics of the underlying support set can be obtained. The works referred here cannot be directly used for our purpose as they do not provide us with the opportunity to group the transactions and find the unique characteristics. But a key point is that the itemsets which are not frequent in nature may not have significant grouping capability. So, to group the transactions, determination of frequent itemsets is important. Any efficient frequent itemset generation technique thus can be used in our case but using level-wise candidate-generation-and-test-based approaches provides us with some advantages as will be discussed in Chapter 4. In fact, we have used the Apriori Algorithm [1] in our work.

### 2.3.2 Sequential Pattern Mining

In the field of sequential pattern mining or frequent subsequence mining, the first motivating work was done by Agrawal and Srikantha in [5]. Since then, a huge number of works were performed in this field. The algorithms developed in this field can broadly divided into two major categories. [5] [6] [77] [78] [79] [80] [81] are in the first category that followed Apriori-based (candidate-generation-and-test-based) approaches and [7] [8] are in the second category that followed pattern growth algorithms to efficiently find frequent subsequences.

**Limitations:** The works in this field of frequent subsequence mining also possess the same limitations for solving our problem as the case discussed for frequent itemset mining. These works also cannot be applied to find the unique characteristics of the underlying support set through the grouping of records or transactions. For this reason, they cannot fulfill the goal of this thesis. But
finding frequent subsequences using one of the above techniques can be used as a part of finding the distinguishing characteristics of the support set.

2.3.3 Compressed Representation of the Patterns

To avoid the generation of redundant patterns, it is necessary to identify only those patterns which are important and relevant, and represent them. There are two types of compressed representation techniques that are used in practice. One is maximal representation of the patterns as followed by [3] [59] [82] where those patterns are selected which do not have frequent superset. Another one is closed representation of the patterns as followed by [83] [84] [85] [86] [87] [88] [60] [89] [90] [91] [92] etc. where those patterns that do not have any frequent superset with the same support are shown in the output.

Limitations: These compressed representation techniques provide the summarized version of the obtained patterns. For this purpose, they omit the redundant patterns and select a pattern from every set of frequent patterns which are related by subset-superset relation. This type of summarized representation of the patterns is not desired for our work. In fact, as the target of this thesis is to group the transactions where a group is formed by a pattern, each pattern is important for us. Also, it is not possible to extend these compressed representation techniques to fulfill our purpose.

2.3.4 Constraint Based Pattern Mining

There is another field of works related to frequent pattern mining which is the constraint based pattern mining. This field aims to build a generic system to solve different pattern mining problems based on different constraints with a minimal effort. Several approaches [93] [94] [95] [96] [97] [98] [99] have been followed to solve a problem through satisfying the constraints. Another similar and related field to constraint based pattern mining is the incorporation of constraint programming in pattern mining as used by [100] [101].

Limitations: Constraint based pattern mining techniques and constraint programming techniques try to provide a system that can be used to find different types of patterns, their compressed representations etc. in such a way that different varieties of purposes can be solved through solving different
constraints without a huge change in the system. Unfortunately, these constraint based pattern mining techniques cannot be used to identify interesting characteristics of the grouped records which should be done in pattern level. Though the purpose of constraint based pattern mining and constraint programming is different from our thesis, the approximate solutions provided by some of the declarative constraint pattern mining techniques can be exploited for our purpose. As for example, [102] provides some heuristics for diverse pattern mining that can be applied in the final phase of our work. But in general, this work is not similar to us as this one aims for determining minimum and diverse pattern set from the raw database which is not our target.

The objective of this thesis is completely different from the works described above. The target of this thesis is to find out the groups of records using pattern level grouping so that the set of records that have same unique characteristics can be grouped together. This challenging task cannot be fully solved in a straight forward way using any of the above-mentioned techniques.

2.4 Summary

In this chapter the background required to understand the problem of this thesis is discussed in brief. Frequent pattern mining concept, their applications and existing works with their limitations to solve the problem of this thesis have been discussed. In the next chapter, the full problem formulation will be discussed where the concepts presented here will work as the background knowledge.
Chapter 3

Problem Formulation

This chapter starts with the definitions of different terms which are required to understand the overall problem of this work. It also defines the new concept *Endemism* along with different notions of *Endemism*. Finally, it provides the original and relaxed version of the problem formulation.

The basic definitions of the related terms will be found in Section 3.1. This section also defines and briefly describes *Endemism*. Then Section 3.2 states the concrete problem formulation along with its relaxed version.

3.1 Definitions of the Related Terms

The underlying support set or the underlying transactions of the data are generally considered as the database of the problem. This database consists of some rows. Each row is called an instance or a record (or a transaction in case of the transaction database). Each record consists of a set of elements. There can be ordering among the elements or the elements can be unordered. In case of frequent itemset mining, each element is a singleton item and no ordering among these items are considered. On the other hand, for sequential pattern mining, each element itself is an unordered set of one or more items but there is a strict ordering among the elements of the record.

**Database:** The support set or all the records (transactions) in combine form the global database or simply the database \( D \). So, for all the records to be considered as the part of the mining will obviously be within this database. In general, the database can be defined as
\[ D = \{ R | R \text{ is a valid record} \} \]

**Record:** A sorted or an unsorted list of elements is called a record, \( R \) (also referred to as instance, transaction etc.). Each record is identified by a record id which is unique to this record. In general, a record can be identified as,

\[ R = \{ E | E \in 2^{IS} \}, \]

where \( IS \) is the set of all valid items.

**Element:** Each record consists of one or more elements. An element is generally expressed as \( E_i \) where \( i \) is the position of the element in that record. Position is important for ordered records but not in unordered ones. Each element can contain one or more items. In an ordered record, each element may have one or more unordered set of items whereas, in case of an unordered record, each element consists of only one item. In general, an element can be defined as,

\[ E = \{ i | i \in IS \}, \]

where \( IS \) is the set of all valid items.

**Item:** Item is the building block of an element. An element consists of one or more items. Each Item is identified by a unique identifier.

Frequent itemset mining is one of the most popular frequent pattern mining area where each transaction (record) consists of an unordered set of items. The concepts Itemset and Frequent Itemset are briefly introduced below:

**Itemset:** A set of unordered items is referred to as an itemset. An itemset may contain one or more items. In general, an itemset \( I \) can be defined as

\[ I = \{ i | i \in IS \}, \]

where \( IS \) is the set of all valid items.

Though one may find similarity between Element and Itemset, the concepts are not the same. In fact, the term Element is used to refer to the member of a record while Itemset is mostly used to define any unordered set of items.

**Frequent Itemset:** With respect to a database \( D \) containing records \( R_1, R_2, \ldots, R_n \), if the number of transactions that contain an itemset is more than a predefined threshold, then this itemset is called
frequent itemset. The predefined threshold can be an absolute number or a number relative to the size of the database. This threshold is known as support threshold. Formally, if \( \text{SupportCount}(I) > s \), where \( I \) is the itemset, \( s \) is the support threshold and \( \text{SupportCount}(X) \) indicates the number of transactions that contain the itemset \( X \), then \( I \) is called a frequent Itemset.

Starting from the Apriori [1] algorithm, most of the frequent pattern mining techniques exploit the anti-monotonic property of the itemsets. To get an idea about the anti-monotonic property, first we define coverage.

**Coverage:** In the database \( D \), if a record \( R_i \) contains the itemset \( I \), then we say that the itemset \( I \) has the coverage of the record \( R_i \) and we can formally write that \( \text{"cover}(I,R_i)_D=\text{True}" \) or simply \( \text{"cover}(I,R_i)=\text{True}" \).

**Anti-monotonic Property:** If two itemsets \( I_1 \) and \( I_2 \) are such that \( I_1 \subseteq I_2 \) and if the statement \( \text{"cover}(I_1,R_i)=\text{False}" \) implies that \( \text{"cover}(I_2,R_i)=\text{False}" \), then this property can be referred to as anti-monotonicity property.

Now, we shall discuss the new concept Endemism. The term Endemism is hugely used in Biology. An animal or a plant is endemic to an area means that this animal or plant is mostly found is that certain area and rare in other areas. For example, Kangaroo is an endemic animal which is found only in Australia continent. This concept can be used herein determining a special type of frequent patterns.

**Endemism:** A pattern (itemset) is referred to as endemic pattern (itemset) if it is most likely to occur in a group rather than individually. That means, if a pattern (itemset) is endemic, then all the elements (items) of the patterns will be found only in the records that this pattern cover, and none of the elements (items) will be found outside of those records. Thus, this pattern (itemset) will be endemic to those records and this characteristic will uniquely separate the records from other records.

A pattern (itemset) can be pure endemic or approximately endemic. So, a score is given to determine the endemism property of that pattern (itemset).

**Pure Endemism:** If it is the case that all the elements (items) of a pattern (itemset) either occur in a certain record or none of them occurs, then the pattern (itemset) is called pure endemic pattern (itemset). If a pattern (itemset) is purely endemic in nature, then it is a must that none of the elements (items) of the patterns (itemsets) will exist outside the group. If the records are grouped based on
pure endemic patterns (itemsets), then there will be no-overlap among the groups of records.

**Approximate Endemism:** If the elements (items) of a pattern (itemset) do not show pure endemism but the elements (items) are most likely to be found with other elements (items) of the pattern (itemset) rather than to be found outside individually, then we can call this pattern (itemset) as an approximate endemic pattern (itemset).

### 3.2 Problem Statement

Our definition of the problem can be divided into two major parts. The first one is to identify the endemism properties of the patterns (itemsets) and the second part is to group the transactions. Formally, the problem can be defined as:

**Problem Definition:** Given a database $D$ consisting of records $R_1, R_2, \ldots, R_m$, the problem is to determine the sets of frequent patterns (itemsets) that divide the underlying records into different groups such that the differentiating patterns (itemsets) are purely endemic to the respective groups and there is no overlap among the groups of records.

**Example:** Consider a sample transaction database where each record is a transaction consisting of a list of items bought together as shown in Table 3.1.

<table>
<thead>
<tr>
<th>Transaction ID</th>
<th>Item List</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R_1$</td>
<td>{A, B, C}</td>
</tr>
<tr>
<td>$R_2$</td>
<td>{X, Y, Z}</td>
</tr>
<tr>
<td>$R_3$</td>
<td>{A, B, C}</td>
</tr>
<tr>
<td>$R_4$</td>
<td>{X, Y, Z}</td>
</tr>
<tr>
<td>$R_5$</td>
<td>{A, B, C}</td>
</tr>
</tbody>
</table>

Table 3.1: Sample Transaction Database

Here in a transaction, either all the items of the itemset \{A, B, C\} co-occur or none of them occur. So the itemset shows pure endemism. This is true for itemset \{X, Y, Z\} too. Based on these two pure
endemic itemsets, we can group the transactions into two groups \( \{R_1, R_3, R_5\} \) and \( \{R_2, R_4\} \). These are our desired groups of records as there are no overlap among the elements of the two groups.

As in the real datasets, it is almost obvious that many of the records will not contain any endemic pattern (itemset) and also the groups formed may be overlapping, so the problem can be relaxed in the following manner.

**Relaxed Problem Definition:** *Given a database \( D \) consisting of records \( R_1, R_2, ..., R_m \), the problem is to determine the sets of \( k \) frequent patterns (itemsets) that divide the underlying records into \( k \) number of groups such that the differentiating patterns (itemsets) are approximately endemic to the respective groups and the overlaps among the groups of records are as less as possible.*

### 3.3 Summary

The definitions of the problem along with the required related terms were the focus of this chapter. To solve the problem, we have proposed a framework which consists of some algorithms. The methodological overview of the thesis along with the detail discussion of the algorithms will be provided in the next chapter.
Chapter 4

Methodologies

This chapter provides the brief description of the methodological details followed in this thesis. Frequent itemsets are considered here as the representative of frequent patterns. The whole procedure of this thesis can be divided into three major tasks: determination of frequent itemsets, scoring the endemic property of the itemset and identification of $k$ number of groups. For the first task, the existing Apriori Algorithm[1] was used. But for the other two tasks there was no existing technique and so new algorithms have been proposed for these. Through the clever use of Apriori Algorithm, the first two tasks can be merged into one - frequent itemset generation and scoring of their endemism. For scoring the endemism, two scoring criteria have been proposed - Reluctancy Scoring and Affinity Scoring. To form the $k$ number of groups, the $k$ number of itemsets are to be selected from $n$ number of candidate frequent itemsets. For this purpose, the ideal strategy is to check all the $\binom{n}{k}$ combinations. This Combinatorial approach forms the baseline algorithm but due to its huge time complexity, it is practically infeasible to use. That is why three heuristics are introduced: TopK Selection, Optimized Search and Random Selection.

The first section (Section 4.1) of this chapter partitions the overall procedure of this work and briefly describes all the parts. Then Section 4.2 discusses the itemset generation and scoring technique, candidate generation process and two types of endemism scoring strategies (Reluctancy Scoring and Affinity Scoring). Section 4.3 is dedicated to the identification of $k$ number of groups through the determination of $k$ number of itemsets. For this subtask, the brief description of the baseline algorithm (Combinatorial approach) and of three heuristics (TopK Selection, Optimized Search, and
Random Search) have been given. This section also includes the Diversity Scoring algorithm and Average Reluctance determination algorithm which are required to evaluate the performances of these different techniques to determine \( k \) number of itemsets.

### 4.1 Proposed Technique to Find Endemic Patterns

The endemic itemsets which will eventually contribute to the grouping of the records must be frequent itemsets. Itemsets which have failed to satisfy a support threshold constant are not frequent in quantity and thus have little significance in determining the groups of records. So the determination of frequent itemsets is a part of our work. The next task is to give an endemism score to all of these frequent itemsets. To give the endemism score to the itemsets, two scoring criteria can be used: Reluctancy Scoring and Affinity Scoring. These scores indicate the endemism property of the corresponding itemset. Finally, based on the endemic characteristics or the endemism scores of the itemsets, we select \( k \) final itemsets that form \( k \) groups of records. So the whole procedure can be divided into three parts. Determination of the frequent itemsets, scoring their endemism and identification of \( k \) groups of records based on the selected \( k \) frequent endemic itemsets.

**Algorithm 4.1 Summarized Algorithm**

1. Determine the list of frequent itemsets, \( F \) using support threshold, \( s \)
2. \textbf{for} all frequent itemset \( c \) in \( F \) \textbf{do}
3. \hspace{1em} Score the endemism of the Itemset, \( c \)
4. \hspace{1em} \textbf{end for}
5. Select the list of top \( k \) number of itemsets, \( T_k \) from \( F \) based on the endemism score
6. Group the records covered by \( T_k \)

The whole procedure is summarized in Algorithm 4.1. In this algorithm, Line No 1 depicts our first task: determination of frequent itemsets, Line 2 to 4 are for the second task: scoring the endemism, and Line numbers 5 and 6 depict the task of identification of \( k \) groups of records.

**Determination of Frequent Itemsets:** Given a support threshold, the task of the identification of the itemsets which satisfy the threshold is a naive one in the field of frequent itemset mining. A lot of works have been performed to efficiently identify the frequent itemsets. We exploit the opportunity to reuse the existing code. In our proposed method, we use the basic candidate generation technique...
used by Apriori Algorithm [1]. The reason to choose this specific algorithm will be discussed very soon.

**Scoring the Endemic Property of the Itemset:** The second task is to score the endemic property or the endemism of the obtained frequent itemsets. This score indicates how endemic the itemsets are. Reluctancy Scoring and Affinity Scoring can be used for scoring the endemism of the itemsets.

**Identification of k Number of Groups:** Based on the endemism score of the itemsets, we can select k number of itemsets among the candidate itemsets so that the k groups of records which would be formed based on these k itemsets produce as less overlap as possible among themselves.

In our proposed method, we have used a small trick. We merge the first two tasks, i.e., we score the endemic property of each frequent itemset during its generation. That is why we have used Apriori Algorithm [1] because Apriori makes a level-wise traversal for frequent itemset generation which is convenient for us to calculate the endemism score as soon as the itemset is identified as the frequent itemset.

### 4.2 Frequent Itemset Generation and Scoring

#### 4.2.1 Frequent Itemset Generation and Scoring Algorithm

As described in the previous section, we combine the first two tasks, i.e., "Determination of Frequent Itemsets" and "Scoring the Endemic Property of the Itemset" in a single method to make the things simpler. For frequent itemset generation purpose, the original Apriori Algorithm [1] has been used here. This algorithm provides us with ample scope to make necessary modifications required to incorporate the scoring opportunity immediately after the identification of an itemset as frequent itemset. The details of the combined algorithm is given as Algorithm 4.2.

In Line 1 of 4.2, the large 1-itemsets are generated. Large 1-itemsets means the single items which fulfill the minimum support threshold. A loop (Line 2-18) makes a level-wise traversal in the enumeration tree of the frequent itemsets. For each level k, it first generates the candidates for the level k from the frequent itemsets of the level (k-1) using the Algorithm 4.3. Then it calculates the support count of all the candidates of level k using each of the records of the database (Line 4-9).
Algorithm 4.2 Frequent Itemset Generation and Scoring Algorithm

1: \( L_1 \leftarrow \{\text{large 1-itemsets}\} \)
2: for \( (k = 2; L_{k-1}; k++) \) do
3: \( C_k \leftarrow \text{candidate\_generate}(L_{k-1}) \)
4: for all records \( r \in D \) do
5: \( C_r \leftarrow \text{subset}(C_k, r) \)
6: for all candidates \( c \in C_r \) do
7: \( \text{c.count}++ \)
8: end for
9: end for
10: \( L_k \leftarrow \{\} \)
11: for all candidate \( c \in C_k \) do
12: if \( \text{c.count} \geq \text{minsup} \) then
13: \( \text{c.reluctancy} \leftarrow \text{getReluctancy}(c, L_1) \)
14: \( \text{c.instanceList} \leftarrow \text{getCoveringInstances}(c, D) \)
15: \( L_k \leftarrow L_k \cup c \)
16: end if
17: end for
18: end for
19: \( P \leftarrow \{\} \)
20: for \( (i = 2; i < k-1; i++) \) do
21: \( P \leftarrow P \cup L_i \)
22: end for
23: return \( P \)

After that, from all the candidate frequent itemsets, the itemsets having support count more than the minimum support threshold are selected as the frequent itemsets of level \( k \). In this step, along with the selection of the frequent itemsets, the reluctance scoring (endemism scoring) of the itemsets are performed, and for all the selected itemsets, the list of the record ids that contain the itemset is prepared. Note that, between the two endemism scoring criteria, reluctance scoring is the best one. The detailed reason and proof of why reluctance scoring is the best will be provided later in this chapter and in the next one. For reluctance scoring, the 4.4 are used in our proposed method. This set of tasks is done for each level \( k \) up to the level where no candidate itemsets can be generated. After the determination of all frequent itemsets, the final task is to accumulate the frequent itemsets of length two or more into a list of frequent itemsets (Line 20-22).
4.2.2 Candidate Generation Algorithm

The Algorithm 4.3 describes the candidate generation process for a particular level $k$, along with pruning strategy used by Apriori [1]. The loop from Line 2 to Line 11 generates all possible candidate itemsets for level $k$. All possible combinations of two itemsets from the frequent itemsets of the level $(k-1)$ are considered. For any combination of two itemsets, if they contain same first $(k-2)$ items and the last item of the first itemset is smaller than the last item of the second itemset, then they form a candidate itemset for level $k$ consisting of all the $(k-2)$ common items, the last item of first itemset and the last item of the second itemset. Note that, for this procedure, an ordered numbering of the items is considered to compare them.

All the generated candidate itemsets for level $k$ may not be promising at all. So an early pruning can delete all the unpromising candidates from the set of candidate itemset list. The pruning is done in Line 13 - 19 in Algorithm 4.3. In this case, the strategy is to prune all the candidate itemsets whose any subset of length $(k-1)$ is not a frequent itemset. The reason is that if any itemset is not frequent, then its superset must not be a frequent itemset too. This property is called the anti-monotonic property which has been discussed in the last chapter.

4.2.3 Endemism Scoring Algorithm: Reluctancy Scoring

A scoring mechanism is required to get an idea of how effective an itemset would be to form a group of records. This scoring actually quantifies the endemism property or the endemic characteristic of the itemset. For scoring the endemism property, we mainly use reluctance scoring mechanism. The key characteristic of the reluctance scoring mechanism is to use the degree of reluctance of each item in the itemset. The degree of reluctance of an item means the tendency of not co-occurring with the other items of the itemset. It can be calculated through dividing the number of records that contain the item but not all the items of the itemset, by the total number of records that contain the item. The degree of reluctance of each item is raised to the power of the size of the itemset and all such values are summed to get the reluctance score of the itemset.

**Degree of Reluctance:** With respect to a pattern (itemset), the degree of reluctance of an element (item) of a pattern (itemset) is the tendency of not co-occurring with all other elements (items) of the
Algorithm 4.3 Candidate Generation

1: \( \text{\textbackslash method: candidate generation for level } k \) 
2: \( \text{\textbackslash for (i = 0; i < size(L_{k-1}); i++) do} \) 
3: \( p \leftarrow i^{th} \text{ member of } L_{k-1} \) 
4: \( \text{\textbackslash for (j = i + 1; j < size(L_{k-1}); j++) do} \) 
5: \( q \leftarrow j^{th} \text{ member of } L_{k-1} \) 
6: \( \text{if } p.item_1 = q.item_1 \text{ and ... and } p.item_{k-2} = q.item_{k-2} \text{ and } p.item_{k-1} < q.item_{k-1} \text{ then} \) 
7: \( c \leftarrow p.item_1 \cup p.item_2 \cup ... \cup p.item_{k-1} \cup q.item_{k-1} \) 
8: \( C_k \leftarrow C_k \cup c \) 
9: \( \text{end if} \) 
10: \( \text{end for} \) 
11: \( \text{end for} \) 
12: \( \text{\textbackslash Pruning for level } k \) 
13: \( \text{\textbackslash for all } c \in C_k \text{ do} \) 
14: \( \text{\textbackslash for all } (k-1)\text{-subsets } x \text{ of } c \text{ do} \) 
15: \( \text{if } x \notin L_{k-1} \text{ then} \) 
16: \( \text{delete } c \text{ from } C_k \) 
17: \( \text{end if} \) 
18: \( \text{end for} \) 
19: \( \text{end for} \) 
20: \( \text{return } C_k \) 

Pattern (itemset) or the reluctance in forming the group with other elements (items) of the pattern (itemset). For a database \( D \), the degree of reluctance of an element (item) \( m \) for the pattern (itemset) \( p \) is expressed as \( \delta(m, p)_D \) or simply \( \delta(m, p) \). In \( D \), if the element (item) occurs in \( y \) times in total and \( x \) times with all other elements (items) of the patterns (itemsets), then probability, \( P \) of finding this element (item), \( m \) as a part of the pattern (itemset), \( p \) is

\[
P(m, p) = \frac{x}{y}
\]

And degree of reluctance, \( \delta \) of the element (item), \( m \) in forming the pattern (itemset), \( p \) is

\[
\delta(m, p) = 1 - P(m, p)
\]

or, \( \delta(m, p) = 1 - \frac{x}{y} \)

**Reluctancy Score:** Reluctancy score is defined at the pattern (itemset) level. The reluctance score of a pattern (itemset) \( p \) for a database \( D \) is expressed as Reluctancy\( (p, D) \) or simply Reluctancy\( (p) \). Reluctancy score is the combination of the degree of reluctances of all the members of the pattern
(itemset). The combination can be generated in many ways but the one that would be used in this thesis is

\[
\text{Reluctancy}(p) = \sum_{m \in p} \delta(m, p)^{|p|},
\]

where \(|p|\) is the size of the pattern (itemset).

To measure the endemism, we use reluctance score. The reluctance score of a pure endemic pattern is 0. The less the reluctance score, the more the endemism shown by the pattern.

Example: Consider a database \(D\), where we want to calculate the endemism of the itemset \(I = \{A, B, C\}\) through their reluctance score. Suppose that, A, B and C appear in this database 10, 9 and 8 times respectively, and their number of occurrences as a group is 5. So, the degree of reluctance of all the items are

\[
\begin{align*}
\delta(A, I) &= 1 - \frac{5}{10} = 0.5 \\
\delta(B, I) &= 1 - \frac{5}{9} = 0.444 \\
\delta(C, I) &= 1 - \frac{5}{8} = 0.375
\end{align*}
\]

As the size of the itemset, i.e., the number of items in the itemset is 3, so the reluctance score of this itemset \(I\) is

\[
\text{Reluctancy}(I) = (0.5)^3 + (0.444)^3 + (0.375)^3 = 0.265
\]

The reluctance scoring technique described here has some advantages. For example, it provides a general reluctance score unbiased of the length of the itemset. The Algorithm 4.4 provides the overview of the scoring strategy. In this case, the degree of reluctance of all the item \(i\) of itemset \(c\), \(\delta(i, c)\) are calculated and summed (Line 6 - 10) to calculate the overall reluctance score. In many cases, the logarithmic value of the calculated reluctance is taken to avoid the problem of too small value.

4.2.4 Alternative Endemism Scoring Algorithm: Affinity Scoring

An alternative technique can be used to score the endemism property of the patterns (itemsets). This scoring strategy is based on taking the ratio of participation of each element (item) of the pattern
Algorithm 4.4 Endemism Scoring Algorithm: Reluctancy Scoring

1: \( c \leftarrow \text{itemset} \)
2: \( L_1 \leftarrow \text{support count of each single frequent item} \)
3: \( s \leftarrow \text{size}(c) \)
4: \( x \leftarrow c.\text{count} \)
5: \( \text{reluctancy} \leftarrow 0 \)
6: \textbf{for all} \( i \in c \) \textbf{do}
7: \( y_i \leftarrow \text{Support count of } i \text{ in } L_1 \)
8: \( \delta(i, c) \leftarrow 1 - \frac{x}{y_i} \)
9: \( \text{reluctancy} \leftarrow \text{reluctancy} + \delta(i, c)^s \)
10: \textbf{end for}
11: \text{return } \log(\text{reluctancy})

The ratio of participation of an element (item) is calculated through dividing the support count of the combined appearance of all the elements (items) of the pattern (itemset) by the individual support count of an element (item) in the database. The logarithmic value of the products of the ratio of participations are taken as the affinity score of the pattern (itemset).

**Ratio of Participation:** With respect to a pattern (itemset), the ratio of participation of an element (item) of the pattern (itemset) is the tendency of forming or the interest in appearing as a group with other elements (items) of the pattern (itemset). It means the tendency of co-occurring with the other elements (items) of the pattern (itemset). For a database \( D \), the ratio of participation of an element (item) \( m \) for the pattern (itemset) \( p \) is expressed as \( \rho(m, p)_D \) or simply \( \rho(m, p) \). In \( D \), if the element (item) occurs in \( y \) times in total and \( x \) times with all other elements (items) of the pattern (itemset), then the ratio of participation is

\[
\rho(m, p) = \frac{x}{y}
\]

The affinity score of a pattern (itemset) \( p \) for a database \( D \) is expressed as \( \text{Affinity}(p, D) \) or simply \( \text{Affinity}(p) \). The more the affinity score, the more endemic the pattern (itemset) is. The affinity score of a pattern (itemset) is determined using the following formula.

\[
\text{Affinity}(p) = \prod_{m \in p} \rho(m, p),
\]

i.e.,

\[
\text{Affinity}(p) = \frac{\text{SupportCount}(p)^{\text{size}(p)}}{\prod_{l \in p} \text{SupportCount}(l)},
\]
Here, $\text{SupportCount}(p)$ means the combined support count of the pattern (itemset), $\text{SupportCount}(i)$ means individual support count of an element (item) of the pattern (itemset), and $\text{size}(p)$ indicates the length of the pattern (itemset).

**Example:** Consider a database $D$, where we want to calculate the endemism of the itemset $I = \{A, B, C\}$ through their affinity score. Suppose that, A, B and C appear in this database 10, 9 and 8 times respectively, and their number of occurrences as a group is 5. So, the ratio of participations of the items are

$$\rho(A, I) = \frac{5}{10} = 0.5$$
$$\rho(B, I) = \frac{5}{9} = 0.555$$
$$\rho(C, I) = \frac{5}{8} = 0.625$$

As the size of the itemset, i.e., the number of items in the itemset is 3, so the affinity score of this itemset $I$ is

$$\text{Affinity}(I) = 0.5 \times 0.555 \times 0.625 = 0.173$$

The procedure to determine the affinity score of an itemset is given in Algorithm 4.5. The heart of the algorithm is the loop from line 5 to line 8. During each iteration of the loop, the value of the numerator and denominator corresponding to the ratio of participation of each item is considered in Line 6 and 7 respectively. Finally, the affinity score is obtained in Line 9. Using the logarithmic value can help to avoid the problem of the resulting small value.

**Algorithm 4.5** Alternative Endemism Scoring Algorithm: Affinity Scoring

1:     num ← 1
2:     denom ← 1
3:     size ← Itemset.size()
4:     combinedSupport ← Itemset.getCombinedSupport()
5:     for all item i ∈ Itemset c do
6:         num ← num*combinedSupport
7:         denom ← denum*getIndividualSupport(i)
8:     end for
9:     return $\log \frac{num}{denom}$

Though affinity scoring can be used for quantifying the endemism of itemset, experiments have proved that reluctance scoring performs better than affinity scoring. The details of the experiments
will be discussed in the next chapter. For now, we shall consider reluctance scoring for our purpose. Obviously, one may consider affinity scoring mechanism instead of reluctance scoring with making necessary changes to the algorithms discussed next.

4.3 Identification of \(k\) Number of Groups

Given a group limit \(k\), the task is to identify at least \(k\) number of groups of records (instances). As each group would be determined by an itemset, the determination of \(k\) number of itemsets from the candidate frequent itemsets is required here to form \(k\) number of groups of records. During the selection of \(k\) number of itemsets, the endemism properities of the itemsets and the overlaps between the groups of records, identified by itemsets, must be considered. So, the task can be specified as the identification of \(k\) frequent itemsets such that 1) each frequent itemset has as much endemism characteristic as possible (as less reluctance score as possible), and 2) the overlaps between the groups of records are as less as possible. Approaches for this selection of \(k\) number of itemsets are discussed in the next subsections.

4.3.1 Baseline Algorithm

It is always better to use an approach which makes the complete search within the solution space. Searching the full solution space is guaranteed to give the optimal solution. In this case, to identify the \(k\) number of itemsets from \(n\) number of candidate itemsets, we can deploy such full searching technique. We can generate all possible \(k\) combinations from the \(n\)-sized candidate set and choose the best one. This Combinatorial approach thus guarantees the optimal solution to the problem. So, this is the baseline algorithm for us. This strategy is depicted in Algorithm 4.6.

First of all, the number of available candidate frequent itemsets, \(n\) and the number of groups to be selected, \(k\) are considered (Line 1 and 2). Then for each of the \(\binom{n}{k}\) combination, diversity score and average reluctance of the combination are determined (Line 5 - 13). The diversity score is the measure of how much diversity are there among the groups of records selected \(k\) itemsets, i.e., how much overlaps among the \(k\) groups of records. The more the diversity score, the less the number of overlaps and the better. On the other hand, average reluctance is the average of the reluctance
scores of the selected combination of itemsets. The algorithm to determine diversity score is given in Algorithm 4.11 and the algorithm to calculate average reluctance is given in Algorithm 4.12. Then the diversity ratio of each combination is determined. The diversity ratio is the ratio of diversity score and average reluctance (as shown in Line 8). Among all combinations, the combination that gives the best diversity ratio is determined. The combination which generates the best diversity ratio is the intended $k$ itemsets for our grouping.

**Algorithm 4.6 Combinatorial Algorithm**

1. $n \leftarrow \text{size}(P)$
2. $k \leftarrow \text{numberOfGroupToBeSelected}$
3. $\text{bestRatio} \leftarrow \text{negative Infinity}$
4. $\text{bestComb} \leftarrow \text{negative Infinity}$
5. **for all** combination $C_b \in \binom{n}{k}$ **do**
6. $d \leftarrow \text{diversityScore}(C_b, I_G)$
7. $\text{avgReluctancy} \leftarrow \text{getAverageReluctancy}(C_b)$
8. $\text{diversityRatio} \leftarrow \frac{\text{diversityScore}}{\text{avgReluctancy}}$
9. **if** $\text{bestRatio} < \text{diversityRatio} \text{ then}$
10. $\text{bestRatio} \leftarrow \text{diversityRatio}$
11. $\text{bestComb} \leftarrow C_b$
12. **end if**
13. **end for**

Combinatorial approach searches the full search space. So it gives the optimal solution. But the problem of the Combinatorial approach is that the huge number of combinations that it generates. For example, if there are 100 candidate frequent itemsets and among these, 50 items to be selected, then the number of combinations is $\binom{100}{50}$ or $\binom{100}{50} \cdot \binom{50}{50}$ which is a huge number that cannot be computed in normal time. That is why we need to consider some heuristics. Three such heuristics are discussed in the next three subsections.

### 4.3.2 Proposed Heuristic 1 (TopK Selection)

To avoid the huge time needed by the Combinatorial approach, a simple strategy or a heuristic must be followed. The first heuristic to be considered is TopK Selection. The target of this heuristic is to select the itemsets that give the minimum average reluctance. It does not consider the diversity score rather focuses on this minimum average reluctance to get a larger diversity ratio. The mechanism followed
by TopK is straightforward - sort the itemsets based on the increasing order of the reluctancy score and take the top k itemsets sequentially from this sorted candidate list.

The methodology of the heuristic is briefly described in Algorithm 4.7. At first, the candidate frequent itemsets are sorted according to the reluctancy score (Line 1). Then the loop from Line 3 to 9 makes a traversal into the sorted list of candidates and selects top k itemsets as the member of final selected list. The traversal terminates as soon as the number of selected itemsets (number of groups) reaches the required number of groups limit (Line 6). The diversity score and the average reluctancy score of the selected final list are determined here in Line 10 and Line 11 respectively. Finally, the diversity ratio is calculated in Line 12.

Algorithm 4.7 Heuristic 1 (TopK Selection) Algorithm

1: $C_s \leftarrow \text{sort}(C_s)$ \hspace{1em} \\text{// Sort itemsets based on increasing order of reluctancy score}
2: groupCount $\leftarrow 0$
3: for all itemset $c \in C_s$ do
4: \hspace{1em} groupCount $\leftarrow$ groupCount + 1
5: \hspace{1em} $L_S \leftarrow L_S \cup c$
6: \hspace{1em} if groupCount = groupLimit then
7: \hspace{2em} break
8: end if
9: end for
10: \hspace{1em} $d \leftarrow \text{diversityScore}(L_S)$
11: \hspace{1em} avgReluctancy $\leftarrow \text{getAverageReluctancy}(L_S)$
12: \hspace{1em} diversityRatio $\leftarrow \frac{\text{diversityScore}}{\text{avgReluctancy}}$

4.3.3 Proposed Heuristic 2 (Optimized Search)

The TopK Selection heuristic is very time efficient. But the problem is that it greedily reduces only the average reluctancy, not the diversity score. So we consider the second heuristic named Optimized Search (OS) which subsumes TopK selection, checks some extra options and finally selects the best one among them. As it searches a fixed number of paths or options, it is not as worse in time complexity as Combinatorial. But it is worse than TopK in time complexity. Again, as it determines the best option among TopK and some other options, it never determines worse diversity ratio than TopK. In fact, in most of the cases, it exhibits better performance (diversity ratio) than TopK and
exhibits equal performance (diversity ratio) in all other cases.

The algorithm of Optimized Search (OS) is shown as Algorithm 4.8. First of all, the previously determined frequent itemsets are sorted according to the increasing order of reluctancy score (Line 3). This sorted list is the list of all candidate frequent itemsets for the final selection. Then, the best shared factor is determined among the values \{0.25, 0.5, 0.75, 1.0\} using the function getBestSharedFactor (Line 9). The details of this function is given in Algorithm 4.9. Note that, when the value of best shared factor is set to 1, then it actually behaves like TopK Selection. Again, the number of values and the set of values used for shared factor can be changed both in deterministic or non-deterministic way. This can be done through tuning or can be incorporated in algorithm implementation. Next, for the best shared factor, the topmost itemset in the list are taken to the final selection list. Then using the loop from Line 10 to Line 27, a sequential traversal through the candidate list is made starting from the second itemset. Among the total records that can be grouped by the itemset, the number of the records that are already assigned to a group is determined (Line 13). This number is called shared number. This number is divided by the total number of records that cover the itemset to get the shared ratio. If this shared ratio is less than shared factor (Line 14), then it is taken in to the final selected list (Line 15 - 23), otherwise it is added to the phase1Rejection list (Line 25). The traversal is stopped as soon as the required number of itemsets (number of groups) is found. If after the sequential traversal of the total candidate list, the total number of itemsets that satisfy the shared factor is less than the group count limit (Line 28), then another traversal is made in the phase1Rejected list (Line 29 - 42). During traversing this phase1Rejected list, the itemsets are sequentially taken to the selected list until the number of itemsets (number of groups) meets the group count limit (Line 39). Then, the diversity score of the selected final list and the average reluctance score of the selected list are calculated in Line 44 and 45 respectively. Finally, the diversity score is calculated as the ratio of diversity score to average reluctance (Line 46).

The getBestSharedFactor function is depicted in Algorithm 4.9. For each shared factor in the set \{0.25, 0.5, 0.75, 1.0\}, a loop (Line 3 - 44) iterates. The logic internally followed by this loop is same as the procedure described above (from Line 10 to 46 in Algorithm 4.8). One key difference is that, instead of returning the diversity ratio, the shared factor that provides with the best diversity ratio is determined (Line 40 - 43) and returned (Line 45).
4.3.4 Proposed Heuristic 3 (Random Selection)

To compare the performances of the first two strategies, we need to know how they perform compared to random selection of the itemsets from the candidate frequent itemsets. Again, in some scenarios where time is very much important, random selection can be used because random selection avoids the problem of the sorting the candidate frequent itemsets based on reluctance scores. This is because the sorting stage may cost a lot in case of a huge candidate frequent itemset list. For all these reasons, the third heuristic to be considered exploits the random selection strategy and thus named as Random Selection.

The methodology is briefly described in 4.10. To select $k$ number of itemsets ($k$ number of groups), a loop from Line 2 to 6 iterates $k$ times. Each time a unique random number is generated and from the list of candidate itemsets, that numbered itemset is taken (Line 3). After selecting the required $k$ number of itemsets, the diversity score and average reluctance score of the selected list are calculated in Line no 7 and 8 respectively. Finally, the diversity ratio is calculated in Line 9.

4.3.5 Diversity Scoring Algorithm

To measure the overlapping tendency among the groups of records determined by the finally selected itemsets, we use Diversity Ratio. Diversity ratio is the ratio of diversity score and average reluctance score. This subsection discusses diversity scoring and the next one discusses average reluctance scoring.

Diversity scoring is the measurement of how diverse the groups of records formed by itemsets are or how fewer overlaps are there among the groups of records. The fewer overlaps, the more diversity. To measure the diversity score of a list of itemsets, all pair of itemsets are considered. For each pair, the overlaps between the record groups formed by itemsets are determined and their Jaccard Distance is calculated. We calculate the diversity score as the average Jaccard Distances obtained considering all such pair of itemsets. The methodology is represented as Algorithm 4.11.

For each iteration of the loop (Line 4 - 10), a distinct pair of itemsets is considered. For each pair of itemsets, the number of common records (instances) covered by both itemsets (Line 5) and the total number of records (instances) covered by the itemsets (Line 6) are calculated. The total score
and the total number of overlaps are also calculated. After considering all the pairs, the average score is calculated in Line 11. This average score is the similarity score. The diversity score is just the subtraction of this similarity score from 1, which is shown in Line 12.

4.3.6 Average Reluctancy Determination

The calculation of the average reluctance of a list of itemsets is straightforward as shown in Algorithm 4.12. A loop (Line 3 - 6) iterates for each itemset of the list. For each itemset, the reluctance of the itemset is determined and added to the total reluctance (Line 4). Finally, the total reluctance is divided by the total number of itemsets to get the average reluctance score (Line 7).

4.4 Summary

This chapter presents the details of our proposed framework. A number of algorithms used for this purpose have been discussed here. Several alternatives have been proposed for both endemism scoring and identification of $k$ number of groups. To evaluate and compare the performances of different alternative approaches, it is important to run a huge number of experiments. The next chapter provides the details of the experimental studies performed for this work.
Algorithm 4.8 Heuristic 2 (Optimized Search) Algorithm

1: $L_R \leftarrow \{\}$  // phase I Rejected List
2: $I_G \leftarrow \{\}$  // List of the grouped instances
3: $C_S \leftarrow \text{sort}(C_S)$  // Sort $C_S$ based on reluctance score
4: $I_C \leftarrow \{\}$  // List of the items covered
5: $L_S \leftarrow \{\}$  // List of the finally selected itemsets
6: $c \leftarrow \text{getFirstCandidate}(C_S)$
7: $L_S \leftarrow L_S \cup c$
8: $I_G \leftarrow I_G \cup c$.getInstance()
9: sharedFactor $\leftarrow$ getBestSharedFactor()
10: for (k=2; k < size($C_S$) and groupCount < groupLimit; k++) do
11:   $c \leftarrow \text{getKthCandidate}(C_S)$
12:   $l \leftarrow c$.getInstance()
13:   shared $\leftarrow$ numberOfSharedWithExistingInstances($I_C$)
14:   if shared < sharedFactor * size($c$) then
15:     groupCount $\leftarrow$ groupCount + 1
16:     $L_S \leftarrow L_S \cup c$, $G \leftarrow \{\}$, $1 \leftarrow c$.getInstance()
17:     for all instance $i \in 1$ do
18:       if !contain($i$, $l_C$) then
19:         $I_C \leftarrow I_C \cup i$
20:         $G \leftarrow G \cup i$
21:     end if
22:   end if
23:   $I_G \leftarrow I_G \cup (c, G)$
24: else
25:   $L_R \leftarrow L_R \cup c$
26: end if
27: end for
28: if groupCount < groupLimit then
29:   for all $c \in L_R$ do
30:     groupCount $\leftarrow$ groupCount + 1
31:     $L_S \leftarrow L_S \cup c$, $G \leftarrow \{\}$, $1 \leftarrow c$.getInstance()
32:     for all instance $i \in 1$ do
33:       if !contain($i$, $l_C$) then
34:         $I_C \leftarrow I_C \cup i$
35:         $G \leftarrow G \cup i$
36:       end if
37:     end for
38:     $I_G \leftarrow I_G \cup (c, G)$
39:   if groupCount = groupLimit then
40:     break
41: end if
42: end for
43: end if
44: $d \leftarrow \text{diversityScore}(L_S)$
45: avgReluctancy $\leftarrow$ getAverageReluctancy($L_S$)
46: return $\frac{d}{\text{avgReluctancy}}$
Algorithm 4.9 getBestSharedFactor() Algorithm

1: bestSharedFactor ← -1.0
2: bestValue ← -1.0
3: for all sharedFactor ∈ {0.25, 0.5, 0.75, 1.0} do
4:     for (k=2; k < size(CS) and groupCount < groupLimit; k++) do
5:         c ← getkthCandidate(CS)
6:         l ← c.getInstances()
7:         shared ← numberOfSharedWithExistingInstances(lC)
8:         if shared < sharedFactor * size(c) then
9:             groupCount ← groupCount + 1
10:            lS ← lS ∪ c, G ← { }, l ← c.getInstances()
11:            for for all instance i ∈ 1 do
12:               if !contain(i, lC) then
13:                   lC ← lC ∪ i
14:                   G ← G ∪ i
15:               end if
16:         end for
17:         lG ← lG ∪ (c, G)
18:     else
19:         lR ← lR ∪ c
20:     end if
21: end for
22: if groupCount < groupLimit then
23:     for all c ∈ lR do
24:         groupCount ← groupCount + 1
25:         lS ← lS ∪ c, G ← { }, l ← c.getInstances()
26:         for for all instance i ∈ 1 do
27:            if !contain(i, lC) then
28:                lC ← lC ∪ i
29:                G ← G ∪ i
30:            end if
31:        end for
32:        lG ← lG ∪ (c, G)
33:    if groupCount = groupLimit then
34:        break
35:    end if
36: end for
37: end if
38: d ← diversityScore(lS)
39: avgReluctancy ← getAverageReluctancy(lS)
40: if bestValue < $\frac{d}{avgReluctancy}$ then
41:    bestValue ← $\frac{d}{avgReluctancy}$
42:    bestSharedFactor ← sharedFactor
43: end if
44: end for
45: return bestSharedFactor
Algorithm 4.10 Heuristic 3 (Random Selection) Algorithm

1: groupCount ← 0
2: while groupCount < groupLimit do
3:     c ← RandomSelection($C_S$) \Randomly select a previously unselected item from $C_S$
4:     groupCount ← groupCount + 1
5:     $L_S ← L_S ∪ c$
6: end while
7: $d ← diversityScore(L_S)$
8: $avgReluctancy ← \text{getAverageReluctancy}(L_S)$
9: return $\frac{d}{avgReluctancy}$

Algorithm 4.11 Diversity Scoring Algorithm

1: count ← 0
2: totalScore ← 0
3: totalOverlap ← 0
4: for all itemset pair (p,q) ∈ $L_S$ do
5:     common ← numberOfIntersectedInstances(p,q)
6:     total ← numberOfTotalInstances(p,q)
7:     count ← count + 1
8:     totalScore ← totalScore + $\frac{\text{common}}{\text{total}}$
9:     totalOverlap ← totalOverlap + common
10: end for
11: avgScore ← $\frac{\text{totalScore}}{\text{count}}$
12: return (1 - avgScore)

Algorithm 4.12 Average Reluctancy Algorithm

1: count ← 0
2: totalReluctancy ← 0
3: for all itemset $c$ ∈ $L_S$ do
4:     totalReluctancy ← totalReluctancy + $c$.getReluctancy()
5:     count ← count + 1
6: end for
7: return $\frac{\text{totalReluctancy}}{\text{count}}$
Chapter 5

Experimental Studies

This chapter provides the experimental details of the thesis. The sources and characteristic details of the datasets are discussed here. Then, to evaluate and compare the performances of different approaches, a number of performance measures are introduced. After that, the experimental outputs are shown and analyzed. The results of the experiments performed on various combinations of different number of rows of the datasets, different support count thresholds and different $k$ number of groups are analyzed for different approaches to compare their performances using the performance measures.

The chapter is structured as follows. Section 5.1 provides the description of the datasets and a brief insight into the characteristic details of each of the dataset. Then section 5.2 introduces different performance metrics to be used to evaluate and compare the results obtained by different approaches. The experimental environment setup is given in Section 5.3. The results obtained for different experiments are briefly described in Section 5.4. Finally, Section 5.5 makes a detailed analysis of the results of the experiments.

5.1 The Dataset

To run the experiments, we have collected 3 real datasets. The summary information of all the datasets are given in Table 5.1. The short description of each of the datasets are given below:
5.1.1 Short Description of the Datasets

Real Dataset 1: Instacart Dataset
Instacart Dataset is a famous online grocery transaction dataset released by instacart in 2017 [103]. The original data of the instacart contains almost 49000 data items of 134 categories. As the average number of the original items in each transaction (record) is less than 10, so the dataset is very sparse in nature. To reduce sparsity of the dataset, we converted this dataset and mapped each item to its category. We kept each category only once in a transaction (record).

Real Dataset 2: Online Retail Dataset
The second real dataset used is a transaction dataset which contains all the transactions occurring between 01/12/2010 and 09/12/2011 for a UK-based and registered non-store online retail. The original dataset can be found in [104]. The number of transactions (records) in this dataset is 25901 and on average there are 20 items sold per transaction.

Real Dataset 3: Online Retail Dataset 2
The third one used for our experiment is also a retail dataset. The dataset contains the (anonymized) retail market basket data from an anonymous Belgian retail store. The original dataset can be found in [105]. But we used the converted version obtained from [106]. This dataset consists of 88162 transactions (records) where each transaction contains on average 9 items.

5.1.2 Characteristic Measures of the Datasets
To get idea about different datasets, we have used several characteristic measures to evaluate their characteristics. The datasets vary largely depending on these characteristics. The corresponding value of different characteristic measures of the datasets are given in 5.1. All the characteristic measures used here are briefly discussed below:

- **Number of Records**: Total number of records or instances (transactions in case of transaction dataset) that is contained in the dataset.

- **Sparsity Index**: Sparsity index indicates how sparse the dataset is. To measure the sparsity of the dataset, the whole dataset was considered as a matrix where records are the rows and
distinct items are the columns. Then we determined the sparsity index of this matrix using the formula $SI = \frac{T_{NZ}}{N_R \times N_C}$, where $T_{NZ}$ means total number of non-zero items in the matrix, $N_R$ means number of rows and $N_C$ indicates number of columns. So, the less the SI value of the dataset, the sparser the dataset.

- **Number of Unique Items**: Total number of unique items contained in the dataset.

- **Maximum Support Count**: The maximum number among the support counts of the items in the dataset. In other word, the maximum coverage made by any item in the dataset.

- **Minimum Support Count**: Similarly, the minimum number among the support counts of the items in the dataset

- **Mean Support Count**: The average of the support counts of all the items in the dataset.

- **Standard Deviation of Support Counts**: The standard deviation value of the support counts of the items in the dataset.

- **Average Number of Items in Each Record**: Average of the number of items contained in a record. In other word, the average length of the records of the dataset.

### 5.2 Performance Measures

To measure the quality of the finally selected list of itemsets, some performance measures or metrics are needed to be evaluated. In this work, the performance metrics used are Diversity Ratio, Relative Relevancy and Time. Diversity Ratio is the ratio of Diversity Score and Average Reluctancy. That is why, we are discussing Diversity Score and Average Reluctancy first, and then move to Diversity Ratio.

1. **Diversity Score**: To measure the overlapping tendency among the groups of records determined by the finally selected itemsets, the diversity score is used. The well-known Jaccard
Distance [107] is used here. The Jaccard Distance is calculated for each pair of groups. We first take a pair of itemsets from the list, determine the groups of records that correspond to the itemsets, and then calculate the Jaccard Distance between these two groups. Thus the Jaccard Distance corresponds to all pair of itemsets are determined. The diversity score is the average of these Jaccard Distances. For example, if two groups of records A and B are formed for the itemset pair (a,b), then

\[
\text{Jaccard Similarity, } J(a,b) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}
\]

\[
\text{Jaccard Distance, } d_J(a,b) = 1 - J(A,B) = \frac{|A \cup B| - |A \cap B|}{|A \cup B|}
\]

The diversity score is the average of Jaccard Distances of all such pairs of selected groups.

2. **Average Reluctancy**: This is the average of the reluctance scores of all the finally selected itemsets.

3. **Diversity Ratio**: The overall target is to maximize the diversity score while keeping the average reluctance minimized. This makes the diversity ratio a key performance metric for this study.
as the diversity ratio is the ratio of diversity score and average reluctancy.

\[
\text{Diversity Ratio} = \frac{\text{Diversity Score}}{\text{Average Reluctancy}}
\]

The target is to maximize the Diversity Ratio. The more the Diversity Ratio value, the better the selected itemsets or the groups of records defined by the selected itemsets.

4. **Relative Relevancy**: The diversity scores of the selected lists of itemsets using different approaches may provide small differences among them. Though these differences are small, they are actually significant. The Relative Relevancy shows this significance. For relative relevancy, we consider a list of \( k \) selected itemsets as the standard list. Then the relative relevancy of a certain list of \( k \) selected itemsets is calculated as the percentage of itemsets that are common with the standard \( k \) selected itemsets.

5. **Time Complexity**: Time complexity is the average time (in ms) needed to calculate and return the final selected list of itemsets. Time complexity is a very vital measure here because due to the problem of huge time complexity, the ideal Combinatorial search cannot be applied here.

Based on these performance metrics, the results of the experiments will be evaluated.

### 5.3 Experimental Setup

For the experiments, the following configuration was used:

- **Operating System**: 64-bit windows 7 operating system
- **Processor**: Intel (R) Core (TM) i5-3230M CPU @ 2.60 GHZ
- **RAM**: 4.00 GB (3.88 GB usable)
- **Platform**: Java
- **JDK**: Version 1.8
- **Editor**: Netbeans 8.0
The implemented code\[108\] of the Apriori Algorithm\[1\] to determine the support threshold based frequent itemset mining was used as a part the full implementation of this work.

A number of experiments were performed on each of the datasets. In these experiments, different techniques were used to identify the final list of itemsets from all the frequent itemsets based on the reluctancy score. The four techniques used in the case were:

- **Combinatorial Approach**: If there are \(k\) itemsets to be selected from \(n\) candidates, all the \(\binom{n}{k}\) combinations are considered and the best combination is selected from these. The best combination is the one which generates the highest diversity ratio. The detail algorithm is shown in Algorithm 4.8 in Chapter 4. A key problem of the Combinatorial approach is that it can generate a huge number of combinations that may not be computable in normal time. For this time complexity problem, it is not a good choice for practical applications. So, three heuristics are used instead of this Combinatorial approach.

- **Heuristic1 (TopK Selection)**: The first heuristic to be considered is TopK Selection. The target of this heuristic is to select the itemsets that give the minimum average reluctancy. It does not consider the diversity score rather focuses on this minimum average reluctancy to get a larger diversity ratio. It makes a straightforward sequential search in the candidate list of itemsets sorted based on their reluctancy score. If \(k\) is the required number of groups (number of itemsets) to be selected, then this procedure selects the top \(k\) itemsets from the candidate list sorted in the increasing order of reluctancy score. The detailed algorithm of this heuristic is shown in Algorithm 4.7 in Chapter 4.

- **Heuristic 2 (Optimized Search)**: The second heuristic is called Optimized Search (OS). It is actually an intermediate strategy between the Combinatorial and the TopK. It considers a fixed number of options including the TopK, and determine the best one among these. It searches the reluctancy score-based sorted candidate list sequentially and select those candidate itemsets which contribute to as less possible number of overlaps among the records of the groups assigned by these itemsets. The tolerance of the overlaps is defined by a shared factor. The procedure stops as soon as the desired number of itemsets (number of groups) becomes equal to the group limit threshold. If this condition is not fulfilled after the first traversal of the se-
lected list, then a second traversal is made on the list of itemsets that were rejected during the first phase. This traversal is made sequentially until the number of group limit condition is fulfilled. After the selection of the list of itemsets, the quality (diversity ratio) of the list is calculated. For different shared factors (we consider four fixed shared factors in this work), the whole process is repeated and the shared factor for which the best diversity ratio is obtained, is identified. Using this shared factor, the final list of selected itemsets are obtained. The full details of the algorithm is shown in Algorithm 4.8 in Chapter 4.

- **Heuristic 3 (Random Selection):** The third and last heuristic used here is Random Selection. It avoids the complexity of sorting the candidate frequent itemsets and randomly selects the required number of itemsets. This heuristic runs quicker than other heuristics but due to its random selection strategy, the diversity ratio of the selected list obtained randomly is not satisfactory at all compared to the other heuristics. In this heuristic, if there are \( k \) number of groups (number of itemsets) to be selected, this heuristic randomly selects \( k \) number of itemsets from the candidate itemset list. The detailed algorithm of this heuristic is shown in Algorithm 4.10 of Chapter 4.

5.4 Results

**Comparison of Endemism Scoring Techniques:** The first experiment was performed to make a comparative analysis of the two endemism scoring techniques: Reluctancy Scoring and Affinity Scoring. For this experiment, a large number of itemsets were considered along with their combined support and individual support of the items. The endemic property of each of the itemsets was manually scored. Then some sample subsets of itemsets were taken, and the accuracy of Reluctancy Scoring and Affinity Scoring were determined. For each subset of itemsets, all pair of itemsets were considered, they were scored using both of the scoring techniques, and checked whether the scoring techniques can determine the correct order between these pair of itemsets. For both of the scoring techniques, the number of pairs for which correct answer obtained is calculated, and the percentage of accuracy was determined. The average percentage accuracy (mean percentage accuracy) and the standard deviation among the percentage of accuracies were calculated considering different subsets
of itemsets. The mean percentage accuracy and standard deviation are shown in table 5.2.

<table>
<thead>
<tr>
<th>Measurement</th>
<th>Reluctancy Scoring</th>
<th>Affinity Scoring</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean Accuracy (in percentage)</td>
<td>94.55</td>
<td>85.45</td>
</tr>
<tr>
<td>Standard Deviation</td>
<td>2.07</td>
<td>3.77</td>
</tr>
</tbody>
</table>

Table 5.2: Mean Percentage Accuracies and Standard Deviation using Reluctancy Scoring and Affinity Scoring in Different Experiments

From this experiment, it is obvious that Reluctancy Scoring outperforms Affinity Scoring for scoring the endemic property of the itemsets. Due to the greater performance of Reluctancy Scoring, we used this scoring technique for rest of the experiments.

The results of the following experiments were performed mainly on three real datasets (Instacart Dataset, Online Retail Dataset 1 and Online Retail Dataset 2). A various number of records, support counts and different values for k were considered for each of the dataset, and the performances of different approaches were evaluated.

Comparing Combinatorial Approach with the Heuristics: This experiment was done to compare the performance of Combinatorial approach with the heuristics. Though Combinatorial search is the ideal one for the final selection of the itemsets (groups), its time complexity is so high that it is practically infeasible to run in real cases. This experiment shows the impact of Combinatorial search. The next experiments will compare the performances of the three heuristics excluding Combinatorial approach.

This experiment compares the ideal combinatorial approach with the above-mentioned heuristics. Combinatorial approach generates a huge number of combinations which are not feasible to compute in practical cases. That is why heuristics are used instead of ideal Combinatorial approach. As the time required to calculate all the combinations in Combinatorial approach is exponential in nature, this experiment consists of a small dataset of only 10000 records of the Instacart dataset [103]. For three support counts, 10, 15 and 20 the Combinatorial approach was compared with the heuristics. More support count can be used, but in that case, the number of generated itemsets will be very
insignificant. For each of the support count, the group factor were varied from 0.1 to 0.9 with 0.1 intervals. The group factor indicates the fraction of candidate itemsets that would be selected finally. Again, for Random Selection a number of iterations were run to get the average statistics.

From the Figure 5.1a, it is clear that no heuristic can have more diversity ratio than the Combinatorial approach. This is because Combinatorial considers each combination and makes an exhaustive search in total candidate space. Optimized Search secures the second place in these experiments defeating the rest two. Random Selection has the worst performance in all the cases. Though the combinatorial gives the highest diversity ratio, the rest others are within the comparable zone. On average, the Diversity Ratio obtained by Optimized Search is 1.81%, obtained by TopK Selection is 3.72% and obtained by Random Selection is 5.45% less the highest Diversity Ratio obtained by Combinatorial. These small differences among the Diversity Ratios are found very significant if the corresponding Relative Relevancy scores are analyzed. Figure 5.1b shows the Relative Relevancy scores for all the approaches where the list of $k$ itemsets obtained by Combinatorial approach is considered as the standard one.

If the time complexity is considered, then Combinatorial is the worst one (as shown in Figure 5.1c). In fact, for this bad time complexity, the Combinatorial approach is infeasible to use in practical scenarios and so the next experiments omit Combinatorial approach.

**Comparing Different Heuristics on 15000 records of the Instacart Dataset:** This experiment was performed to get an idea about the performance of different heuristics used as the alternative to the Combinatorial approach. In this case, a number of runs were performed on 15000 records of the Instacart dataset. The support counts were varied from 25 to 70 with 5 interval range. This range was selected because the support value below the range provides unnecessary itemsets that have a rare influence on the final groups of records. On the other hand, making support count more than 70 produces only a few candidate frequent itemsets of length more than 1. A number of iterations were conducted on each of the heuristics to get the average performance measures. Again, for each run, the group factor was varied from 0.1 to 0.9 with an interval of 0.1. The result of this experiment is shown in Figure 5.2. As we did not using Combinatorial approach here and as the Optimized Search
(OS) showed the highest Diversity Ratio and Relative Relevancy after the Combinatorial, we used the list of \( k \) itemsets found by the Optimized Search (OS) as the standard list for calculating the Relative Relevancy score.

**Comparing Different Heuristics on 20000 records of the Instacart Dataset:** This experiment was performed on 20000 records of the Instacart Dataset. In this case, various support counts were varied from 25 to 70 with 5 interval range. This range was selected for the same reason as described above. A number of runs were performed on each of the heuristics to get the average performance measures. Again, for each run, the group factor was varied from 0.1 to 0.9 with an interval of 0.1. Figure 5.3 shows the result of this experiment. Here, we used the list of \( k \) itemsets found by the Optimized Search (OS) as the standard list for calculating the Relative Relevancy score.

**Comparing Different Heuristics on 50000 records of the Instacart Dataset:** For this experiment, the number of taken records was increased to 50000. In this case, various support counts were used as shown in Figure 5.4. For each support count, a number of runs were performed similarly as before to compare the heuristics. For all the heuristics, the average performance was considered. The group factor was similarly varied from 0.1 to 0.9 with an interval of 0.1. We used the list of \( k \) itemsets found by the Optimized Search (OS) as the standard list for calculating the Relative Relevancy score.

**Comparing Different Heuristics on 15000 records of the Online Retail Dataset1:** For this experiment, we used Online Retail Dataset1. The number of records considered were 15000. In this case, various support counts were used as shown in Figure 5.5. For each support count, a number of runs were performed similarly as before to compare the heuristics. For all the heuristics, the average performance was considered. The group factor was varied from 0.1 to 0.9 with an interval of 0.1. We used the list of \( k \) itemsets found by the Optimized Search (OS) as the standard list for calculating the Relative Relevancy score.

**Comparing Different Heuristics on 20000 records of the Online Retail Dataset1:** For this experiment, we used the same Online Retail Dataset1. But the number of records were 20000. In this case, various support counts were used as shown in Figure 5.6. For each support count, a number of runs were performed to get the average performance of the heuristics. The group factor was varied from 0.1 to 0.9 with an interval of 0.1. Here, we used the list of \( k \) itemsets found by the Optimized
Search (OS) as the standard list for calculating the Relative Relevancy score.

Comparing Different Heuristics on 25000 records of the Online Retail Dataset2: This experiment was conducted on Real Dataset 3 (Online Retail Dataset 2). The number of records taken was 25000. Various support counts were used in this experiment as shown in Figure 5.7. For each support count, a number of runs were performed to get the average performance of the heuristics. The group factor was varied from 0.1 to 0.9 with an interval of 0.1. Here, we used the list of k itemsets found by the Optimized Search (OS) as the standard list for calculating the Relative Relevancy score.

Comparing Different Heuristics on 35000 records of the Online Retail Dataset2: 35000 records of Real Dataset 3 (Online Retail Dataset 2) were considered for this experiment. Also various support counts were used as shown in Figure 5.8. For each support count, a number of runs were performed similarly to get the average performance of the heuristics. The group factor was varied from 0.1 to 0.9 with an interval of 0.1. We used the list of k itemsets found by the Optimized Search (OS) as the standard list for calculating the Relative Relevancy score.

5.5 Discussion

The mean diversity ratios obtained by different heuristics in all different experiments are shown in Table 5.3. Similarly, the mean relative relevancy scores and mean execution times of different the heuristics are shown in Table 5.4 and Table 5.5 respectively.

Diversity Ratio: From the graphs shown in the previous section and the data shown in Table 5.3, it is clear that the diversity ratio obtained by Optimized Search is the highest one among the three heuristics. This makes sense because Optimized Search checks for different values of instance shared factor. Among these, the value 1 is actually allowing any kind of sharing, which is indirectly the TopK Selection. That means, the Optimized Search indirectly checks for TopK and some other options, and takes the best one among these. So this would be obviously better than TopK as expected. Again, Random does not take reluctance or overlaps into consideration, so the overall goal of minimizing reluctance while maximizing diversity score cannot be ensured by Random. On the other hand, as TopK tries to minimize the reluctance, its performance is better than Random as expected.

Relative Relevancy: The relative relevancy measure actually shows the significant improvement
of the Optimized Search (OS) and TopK Selection heuristics comparing to the Random Selection heuristic. It is also obvious from the Figure 5.1 and Table 5.4 that Optimized Search selects better list of $k$ itemsets than TopK. In this case, it is also clear that on average Random Selection picks up 50% from the standard $k$ itemsets and 50% from outside of the list. This event explains why Random Selection shows the poor performance.

**Time Complexity:** The execution time taken by the Optimized Search is always greater than the other two approaches as proved by the results of the experiments (Table 5.5). The reason behind this is that Optimized Search sorts the elements and considers several possible options in the search space. So the time taken by it is as expected longer than other ones. The execution time of TopK Selection is always less than Optimized Search. This is because, Optimized search sorts the elements as TopK, do a search as Topk and additionally checks for few more options. In this case, the sorting time and one search time is same for both Optimized and Topk. But for other additional searches and the extra calculations for these searches make Optimized Search slower than Topk. On the other hand, the Random Selection takes the least time. This is obvious because Random avoids the sorting phase of the candidate itemsets which cost a lot for both Optimized Search and TopK.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>No of Records</th>
<th>Optimized Search (OS)</th>
<th>TopK Selection</th>
<th>Random Selection</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instacart:</td>
<td>15000</td>
<td>0.549656</td>
<td>0.544347</td>
<td>0.534664</td>
</tr>
<tr>
<td>Instacart:</td>
<td>20000</td>
<td>0.54555</td>
<td>0.54003</td>
<td>0.53151</td>
</tr>
<tr>
<td>Instacart:</td>
<td>50000</td>
<td>0.53198</td>
<td>0.52678</td>
<td>0.51956</td>
</tr>
<tr>
<td>Online Retail1: 15000</td>
<td>0.704</td>
<td>0.6759</td>
<td>0.6404</td>
<td></td>
</tr>
<tr>
<td>Online Retail1: 20000</td>
<td>0.6847</td>
<td>0.6582</td>
<td>0.6211</td>
<td></td>
</tr>
<tr>
<td>Online Retail2: 25000</td>
<td>0.7573</td>
<td>0.7295</td>
<td>0.5035</td>
<td></td>
</tr>
<tr>
<td>Online Retail2: 35000</td>
<td>0.80792</td>
<td>0.7346</td>
<td>0.5055</td>
<td></td>
</tr>
</tbody>
</table>

Table 5.3: Mean Diversity Ratios in Different Experiments
### CHAPTER 5. EXPERIMENTAL STUDIES

<table>
<thead>
<tr>
<th>Dataset</th>
<th>No of Records</th>
<th>Optimized Search (OS)</th>
<th>TopK Selection</th>
<th>Random Selection</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instacart</td>
<td>15000</td>
<td>100</td>
<td>94.648</td>
<td>49.851</td>
</tr>
<tr>
<td>Instacart</td>
<td>20000</td>
<td>100</td>
<td>95.155</td>
<td>49.138</td>
</tr>
<tr>
<td>Instacart</td>
<td>50000</td>
<td>100</td>
<td>94.209</td>
<td>49.773</td>
</tr>
<tr>
<td>Online Retail1</td>
<td>15000</td>
<td>100</td>
<td>82.359</td>
<td>51.774</td>
</tr>
<tr>
<td>Online Retail1</td>
<td>20000</td>
<td>100</td>
<td>84.612</td>
<td>49.676</td>
</tr>
<tr>
<td>Online Retail2</td>
<td>25000</td>
<td>100</td>
<td>97.289</td>
<td>52.134</td>
</tr>
<tr>
<td>Online Retail2</td>
<td>35000</td>
<td>100</td>
<td>92.191</td>
<td>49.761</td>
</tr>
</tbody>
</table>

Table 5.4: Mean Relative Relevancy Scores in Different Experiments

<table>
<thead>
<tr>
<th>Dataset</th>
<th>No of Records</th>
<th>Optimized Search (OS)</th>
<th>TopK Selection</th>
<th>Random Selection</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instacart: 15000</td>
<td>2.5</td>
<td>1.6</td>
<td>1.2</td>
<td></td>
</tr>
<tr>
<td>Instacart: 20000</td>
<td>7.1</td>
<td>3.3</td>
<td>2.6</td>
<td></td>
</tr>
<tr>
<td>Instacart: 50000</td>
<td>184.1</td>
<td>87.3</td>
<td>62.7</td>
<td></td>
</tr>
<tr>
<td>Online Retail1: 15000</td>
<td>438.4</td>
<td>2.8</td>
<td>2.4</td>
<td></td>
</tr>
<tr>
<td>Online Retail1: 20000</td>
<td>1517.8</td>
<td>58</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>Online Retail2: 25000</td>
<td>25.9</td>
<td>1.9</td>
<td>1.4</td>
<td></td>
</tr>
<tr>
<td>Online Retail2: 35000</td>
<td>85.2</td>
<td>4.2</td>
<td>1.6</td>
<td></td>
</tr>
</tbody>
</table>

Table 5.5: Mean Execution Time Different Experiments
Figure 5.1: Result Showing Comparison of Combinatorial Approach with the Heuristics
Figure 5.2: Result Showing Comparison among Different Heuristics on 15000 records of the Instacart Dataset
(a) Average Diversity Ratios of Different Heuristics

(b) Average Relative Relevancy Scores of Different Heuristics

(c) Average Time needed by Different Heuristics

Figure 5.3: Result Showing Comparison among Different Heuristics on 20000 records of the Instacart Dataset
Figure 5.4: Result Showing Comparison among Different Heuristics on 50000 records of the Instacart Dataset
Figure 5.5: Result Showing Comparison among Different Heuristics on 15000 records of the Online Retail Dataset 1
Figure 5.6: Result Showing Comparison among Different Heuristics on 20000 records of the Online Retail Dataset1
(a) Average Diversity Ratios of Different Heuristics

(b) Average Relative Relevancy Scores of Different Heuristics

(c) Average Time (in logarithmic scale) needed by Different Heuristics

Figure 5.7: Result Showing Comparison among Different Heuristics on 25000 records of the Online Retail Dataset2
Figure 5.8: Result Showing Comparison among Different Heuristics on 35000 records of the Online Retail Dataset2
Chapter 6

Conclusion

6.1 Conclusion

Frequent pattern mining is an established field of data mining which is expanding day by day. To make the future of this field more fruitful, determination of new types of patterns that can extract meaningful information of the underlying support set is a crying need. This thesis is motivated by this need. In this thesis, a new field of frequent pattern mining has been explored where the records of the underlying support set are grouped based on their unique characteristics. This grouping is done using the pattern level uniqueness. All the frequent patterns are determined and a special score is given to them to indicate their endemism, i.e., the unique identifying capability. Between the two given endemism scoring techniques, the Reluctancy Scoring technique outperforms the Affinity Scoring technique almost in all the cases. Then, based on the endemism scoring, a number of patterns are identified that form the groups of records maintaining the less possible overlaps among the groups. Several approaches can be taken to serve the purpose of identifying the final selected patterns. The complete or exhaustive search procedure adopted by Combinatorial approach is the baseline algorithm and provides us with the best possible result. But launching Combinatorial search is practically infeasible due to its exponential time complexity. As alternative, several heuristics are given - TopK Selection, Optimized Search (OS) and Random Selection. Some important performance metrics are also introduced. A number of experiments were performed to evaluate the performances of the heuristics and the outcomes of the experiments were analyzed based on these metrics. Among
the heuristics, the Optimized Search (OS) showed the highest performance though it needed a small sacrifice of time.

6.2 Future Work

This work was the first one, to the best of our knowledge, that aimed to find the unique or distinguishing characteristics of the underlying transactions or support set using the grouping of the records or transactions. We provided a well organized framework to solve the problem. As this was our first work for solving this problem, we have some plans to be executed in the future.

- The methodology described and the experiments done in this thesis considered the unordered databases, i.e., for experiments we considered frequent itemsets only. In future, the ordered databases will be considered as a part of our analysis and we shall work on frequent subsequences.

- To find the $k$ final itemset list, we introduced some heuristics as alternative to Combinatorial approach. Though these heuristics showed very good performances but in future more promising heuristics will be searched so that both the performance and time complexity are within an acceptable range.

- More experiments will be performed on different types of datasets to explore the impact of different datasets on different performance metrics.

- During the experiments, different interesting observations were found, e.g., the change of diversity ratio with the change of support count. We hope to make an extensive research to find the actual reason behind these type of interesting events.

- It will be interesting to investigate how the support count influences the final grouping of records and to recommend a support count value for a dataset that will produce the best groups of records. Also, we shall try to consider percentage value of support count with respect to the number of records of the dataset instead of using absolute support count value.
• We want to try other different candidate-generation-test-based approaches for identification of frequent itemsets, and compare the impact of different approaches for this purpose.

• In this work, we proposed a framework which makes an implicit grouping or clustering of the records. In future, we want to use some existing renowned clustering or grouping methods for the grouping of the records, and compare the performances of different clustering and grouping approaches including the provided one in this work based on the evaluation criteria proposed by us.
Bibliography


[7] Jiawei Han, Jian Pei, Behzad Mortazavi-Asl, Qiming Chen, Umeshwar Dayal, and Mei-Chun Hsu. Freespan: Frequent pattern-projected sequential pattern mining. In *Proceedings of the*
Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD ’00, pages 355–359, New York, NY, USA, 2000. ACM.


[17] Xifeng Yan, Feida Zhu, Jiawei Han, and P. S. Yu. Searching substructures with superimposed distance. In 22nd International Conference on Data Engineering (ICDE’06), pages 88–88, April 2006.


[29] Chao Liu, Xifeng Yan, Hwanjo Yu, Jiawei Han, and Philip S. Yu. Mining Behavior Graphs for Backtrace of Noncrashing Bugs, pages 286–297. 2005.


[37] Hongyan Liu, Jiawei Han, Dong Xin, and Zheng Shao. Mining frequent patterns from very high dimensional data: A top-down row enumeration approach *. In SDM Conference, 2006.


on Knowledge Discovery and Data Mining, KDD '02, pages 429–435, New York, NY, USA, 2002. ACM.


[91] Jianyong Wang, Jiawei Han, and Jian Pei. CLOSET+: Searching for the best strategies for mining frequent closed itemsets, pages 236–245. 2003.


[99] Jian Pei and Jiawei Han. Can we push more constraints into frequent pattern mining? In Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD ’00, pages 350–354, New York, NY, USA, 2000. ACM.


