

An Efficient Approach for High Utility Itemset Mining

A Thesis submitted to Gujarat Technological University

For the Award of

Doctor of Philosophy

in

Computer/IT Engineering

By

Patel Sureshkumar Bhikhabhai

Enrollment No. 189999913029

Under the supervision of

Dr. Sanjay M Shah



**GUJARAT TECHNOLOGICAL UNIVERSITY,
AHMEDABAD**

February - 2024

An Efficient Approach for High Utility Itemset Mining

A Thesis submitted to Gujarat Technological University

For the Award of

Doctor of Philosophy

in

Computer/IT Engineering

By

Patel Sureshkumar Bhikhabhai

Enrollment No. 189999913029

Under the supervision of

Dr. Sanjay M. Shah



**GUJARAT TECHNOLOGICAL UNIVERSITY,
AHMEDABAD**

February - 2024

© PATEL SURESHKUMAR BHIKHABHAI

DECLARATION

I declare that the thesis entitled "**An Efficient Approach for High Utility Itemset Mining**" submitted by me for the degree of Doctor of Philosophy is the record of research work carried out by me during the period from February-2019 to February -2024 under the supervision of **Dr. Sanjay M. Shah** and this has not formed the basis for the award of any degree, diploma, associateship, fellowship, titles in this or any other University or other institution of higher learning.

I further declare that the material obtained from other sources has been duly acknowledged in the thesis. I shall be solely responsible for any plagiarism or other irregularities, if noticed in the thesis.

Signature of Research Scholar:



Date: 12/02/2024

Name of Research Scholar: **Patel Sureshkumar Bhikhabhai**

Place:

Ahmedabad

CERTIFICATE

I certify that the work incorporated in the thesis "**An Efficient Approach for High Utility Itemset Mining**" submitted by **Mr. Patel Sureshkumar Bhikhabhai** was carried out by the candidate under my supervision/guidance. To the best of my knowledge: (i) the candidate has not submitted the same research work to any other institution for any degree/diploma, Associateship, Fellowship or other similar titles (ii) the thesis submitted is a record of original research work done by the Research Scholar during the period of study under my supervision, and (iii) the thesis represents independent research work on the part of the Research Scholar.

Signature of Supervisor:



Date: 12/02/2024

Name of Supervisor: **Dr. Sanjay M Shah**

Place:

GTU Ahmedabad

Course-work Completion Certificate

This is to certify that Mr. **Patel Sureshkumar Bhikhabhai** Enrollment no. **189999913029** is a PhD scholar enrolled for the PhD program in the branch **Computer/IT Engineering** of Gujarat Technological University, Ahmedabad.

(Please tick the relevant option(s))

☐

He has been exempted from the course-work (successfully completed during M.Phil. Course)

☐

He has been exempted from Research Methodology Course only (successfully completed during M.Phil Course)

☒

He has successfully completed the PhD course work for the partial requirement for the award of PhD Degree. His performance in the course work is as follows-

Grade Obtained in Research Methodology (PH001)	Grade Obtained in Self Study Course (Core Subject) (PH002)
BC	AB

Supervisor's Sign

(Dr. Sanjay M. Shah)

Originality Report Certificate

It is certified that Ph.D. Thesis titled "**An Efficient Approach for High Utility Itemset Mining**" by **Patel Sureshkumar Bhikhabhai** has been examined by us.

We undertake the following:

- a. Thesis has significant new work/knowledge as compared already published or are under consideration to be published elsewhere. No sentence, equation, diagram, table, paragraph, or section has been copied verbatim from previous work unless it is placed under quotation marks and duly referenced.
- b. The work presented is original and own work of the author (i.e. There is no plagiarism). No ideas, processes, results or words of others have been presented as Author own work.
- c. There is no fabrication of data or results which have been complied/analysed.
- d. There is no falsification by manipulating research materials, equipment or processes, or changing or omitting data or results such that the research is not accurately represented in the research record.
- e. The thesis has been checked using "**Drillbit Plagiarism Checker**" (copy of originality report attached) and found within limits as per GTU Plagiarism Policy and instructions issued from time to time (i.e. permitted similarity index $\leq 10\%$).

Signature of Research Scholar: 

Date: 12/02/2024

Name of Research Scholar: **Patel Sureshkumar Bhikhabhai**

Place: Ahmedabad.

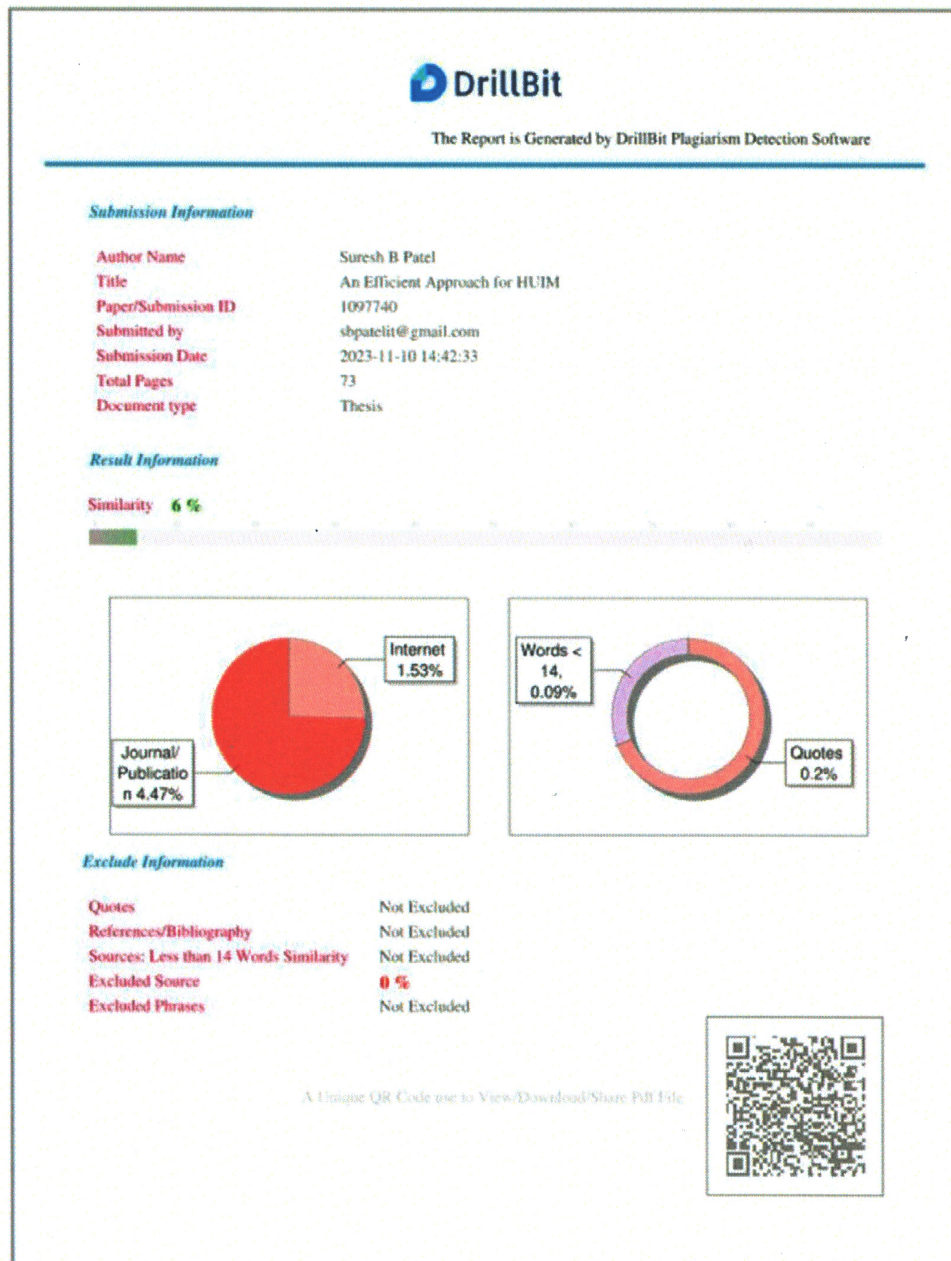
Signature of Supervisor: 

Date: 12/02/2024

Name of Supervisor: **Dr. Sanjay M. Shah**

Place: GTU Ahmedabad

Copy Originality Report





DrillBit Similarity Report

6

SIMILARITY %

6

MATCHED SOURCES

A

GRADE

A-Satisfactory (0-10%)
B-Upgrade (11-40%)
C-Poor (41-60%)
D-Unacceptable (61-100%)

LOCATION	MATCHED DOMAIN	%	SOURCE TYPE
1	www.ncbi.nlm.nih.gov	1	Internet Data
2	Location-Aware Deep Collaborative Filtering for Service Recommendation by Zhang-2019	<1	Publication
3	www.readbag.com	1	Internet Data
4	docview.dlib.vn	4	Publication
5	ACM Press the International Academic Conference- Vancouver, Britis, by Shen, Zhiqi Miao, - 2010	<1	Publication
6	www.dx.doi.org	<1	Publication

PhD Thesis Non-Exclusive License to GUJARAT TECHNOLOGICAL UNIVERSITY

In consideration of being PhD Research Scholar at GTU and in the interests of the facilitation of research at GTU and elsewhere I, "**Patel Sureshkumar Bhikhabhai**" having Enrollment No. **189999913029** hereby grant a non-exclusive, royalty-free and perpetual license to GTU on the following terms:

- a. The University is permitted to archive, reproduce and distribute my thesis, in whole or in part, and/or my abstract, in whole or in part (referred to collectively as the "Work") anywhere in the world, for non-commercial purposes, in all forms of media;
- b. The University is permitted to authorize, sub-lease, sub-contract or procure any of the acts mentioned in paragraph (a);
- c. The University is authorized to submit the Work at any National / International Library, under the authority of their "Thesis Non-Exclusive License";
- d. The Universal Copyright Notice (©) shall appear on all copies made under the authority of this license;
- e. I undertake to submit my thesis, through my University, to any Library and Archives. Any abstract submitted with the thesis will be considered to form part of the thesis.
- f. I represent that my thesis is my original work, does not infringe any rights of others, including privacy rights, and that I have the right to make the grant conferred by this non-exclusive license.
- g. If third party copyrighted material was included in my thesis for which, under the terms of the Copyright Act, written permission from the copyright owners is required, I have obtained such permission from the copyright owners to do the acts mentioned in paragraph (a) above for the full term of copyright protection.
- h. I understand that the responsibility for the matter as mentioned in the paragraph (g) rests with the authors / me solely. In no case shall GTU have any liability for any acts / omissions / errors / copyright infringement from the publication of the said thesis or otherwise.

- i. I retain copyright ownership and moral rights in my thesis, and may deal with the copyright in my thesis, in any way consistent with rights granted by me to my University in this non-exclusive license.
- j. GTU logo shall not be used /printed in the book (in any manner whatsoever) being published or any promotional or marketing materials or any such similar documents.
- k. The following statement shall be included appropriately and displayed prominently in the book or any material being published anywhere: "The content of the published work is part of the thesis submitted in partial fulfilment for the award of the degree of Ph.D. in Computer/IT Engineering of the Gujarat Technological University".
- l. I further promise to inform any person to whom I may hereafter assign or license my copyright in my thesis of the rights granted by me to my University in this nonexclusive license. I shall keep GTU indemnified from any and all claims from the Publisher(s) or any third parties at all times resulting or arising from the publishing or use or intended use of the book / such similar document or its contents.
- m. I am aware of and agree to accept the conditions and regulations of Ph.D. including all policy matters related to authorship and plagiarism.

Date: 12/02/2024

Place: Ahmedabad

Signature of the Research Scholar

Recommendation of the Research Supervisor: RECOMMENDED

Signature of the Research Supervisor

Thesis Approval Form

The viva-voce of the PhD Thesis submitted by **Mr. Patel Sureshkumar Bhikhabhai** (Enrollment No. **189999913029**) entitled " **An Efficient Approach for High Utility Itemset Mining**" was conducted on 12/02/2024 at Gujarat Technological University.

(Please tick any one of the following options)



The performance of the candidate was satisfactory. We recommend that he be awarded the PhD degree.



Any further modifications in research work recommended by the panel after 3 months from the date of first viva- voce upon request of the Supervisor or request of Independent Research Scholar after which viva – voce can be re-conducted by the same panel again.

(Briefly specify the modifications suggested by the panel)



The performance of the candidate was unsatisfactory. We recommend that , he should not be awarded the PhD degree.

(The panel must give justifications for rejecting the research work)



(Dr. Sanjay M. Shah)

Name and signature of Supervisor with Seal



(Dr. Nilesh R Patel)

External Examiner-1 (Name and Signature)



(Dr. Rajiv Ranjan)

External Examiner-2 (Name and Signature)

Abstract

In today's highly competitive business landscape, the need for an effective DSS (Decision Support System) is an essential. Modern business organization has generated a vast amount of raw data related to product, service, customers, etc. A hidden pattern in these raw data is a key to designing an efficient DSS for the organization. Manual extraction of these patterns from the large volume of raw data is unfeasible. Therefore, an association rule mining: a data mining technique is widely used to extract useful pattern from the raw data for designing efficient decision support system. The conventional association rule mining is frequent itemset mining. It is widely used and popular for extracting related items. The traditional approach focuses on whether the group of items frequently appears in the dataset or not. However, in certain real-world scenarios, it becomes essential to consider the quantity and importance of items as well. For example, in a supermarket to identify profitable items from customer transaction data or in the medical field to discover combinations of highly indicative symptoms of diseases. High utility itemset mining incorporates the item's quantity and importance to address this need. Number of research has been carried out on high utility itemset mining, with utility list-based methods emerging as efficient techniques. These methods avoid the generation of candidate sets, which can be computationally expensive. However, a major drawback of existing utility list-based techniques is the need for costly join operations on utility lists. These operations can degrade the algorithm's performance by increasing execution time and storage requirements. The cost of the utility list join operations is directly related to the number of comparisons required to find out the common transactions between the utility lists. In this study, Proposed SCAO (Support Count Ascending Order) based search space exploration technique to decrease the number of comparisons required to find common transactions between the utility lists. So, it can minimize the cost of the join operations.

Also, existing state-of-the-art approaches perform unnecessary utility list join operations of the itemsets that are not high utility itemsets. To address this issue, our proposed PUCP (Predicted Utility Co-exist Pruning) uses PUCS (Predicted Utility Co-exist Structure) to eliminate unnecessary join operations. The proposed approach using the PUCP is called the PUCP-Miner. The performance of the suggested SCAO-based approach and PUCP-Miner is assessed with the existing algorithms like HUI-Miner, mHUI-Miner, and ULB

miner on some of the standard real datasets. The experimental outcomes show that the proposed SCAO-based approach and PUCP-miner have outperformed existing state-of-the-art methods by up to 59% in running time and up to 46% in memory consumption.

Acknowledgement

I want to express my sincere thanks to everyone who has provided unwavering support and encouragement during my doctoral research journey.

I want to begin my heartfelt thanks to Dr. Sanjay M Shah sir, my Ph.D. supervisor. He is a Professor and serves as the Head of the Computer Engineering Department at Government Engineering College, Rajkot. Sir has consistently provided me with invaluable support and thoughtful guidance throughout the entire duration of my research. I am profoundly grateful for his unique insights, continuous motivation, and the significant time he dedicated to shaping this research and illuminating a previously undisclosed aspect of the study.

I would like to extend my sincere gratitude to the members of my Doctorate Progress Committee (DPC), namely Dr. Sanjay P. Patel, who serves as an Assistant Professor in the Computer Engineering Department at Government Engineering College, Gandhinagar, and Dr. Uttam G Chauhan, an Assistant Professor in the Computer Engineering Department at Vishwakarma Government Engineering College, Chandkheda. Their insightful comments, valuable suggestions, and encouragement to view the problem from various angles have been immensely helpful. Their approachable demeanour and appreciation for my work have consistently fostered a supportive atmosphere, boosting my confidence to overcome challenges and push my boundaries.

I am also thankful to the Honourable Vice-Chancellor, Registrar, Controller of Examination, Dean of the PhD section, and the entire team at the PhD Section of Gujarat Technological University (GTU) for their invaluable assistance and unwavering support.

I would like to express my deep appreciation to Dr. S. P. Dave, the Principal of GEC Gandhinagar, Dr. D. A. Parikh, the Head of the CE Department, and Professor J. S. Dhobi from the IT department for granting me the necessary resources and facilities to reach my desired goals.

I wish to extend my gratitude to my parent institution, Government Engineering College, Gandhinagar, and the Department of Technical Education, Gujarat, for their comprehensive support throughout my research journey. I am fortunate to have received blessings and guidance from the Dr. K. K. Jani, and Dr. Pratik Barot. Additionally, I

would like to express my thanks to my colleagues and well-wishers in the CE/IT department for their continuous encouragement and support during the entire duration of my research work.

I hold the utmost respect and affection for my parents, wife, brothers, and my children Siddhi and Aarav, as well as my niece and nephews, Riyan, Preksha, Siddh, Anay, Jency, and Meera. I want to convey my profound gratitude for their unwavering support and collaboration. I am particularly grateful to my wife, Vandana, for her consistent encouragement, tireless support during my research, and her overall cooperation.

Finally, I want to convey my profound gratitude to my dear friend Mahendra N Patel from the depths of my heart. Without his continuous support, motivation, and encouragement, reaching this stage would not have been possible.

Since this has been a lengthy journey, there might be a few names I have unintentionally forgotten, yet they have played a crucial role in this significant undertaking. I humbly ask for forgiveness for any oversight and extend my sincere respect to each and every one of them.

I am extremely thankful to Almighty God for giving me patience and strength to complete this research.

Patel Suresh B.

Table of Contents

DECLARATION	i
CERTIFICATE	ii
Course-work Completion Certificate	iii
Originality Report Certificate	iv
Copy Originality Report	v
PhD Thesis Non-Exclusive License	vii
Abstract.....	x
Acknowledgement.....	xii
Table of Contents	xiv
List of Abbreviations.....	xvi
List of Figures	xviii
List of Tables	xx
Chapter 1. Introduction	1
1.1 Introduction	1
1.2 Background Study	2
1.2.1 Data Mining.....	2
1.2.2 Knowledge Discovery From Database.....	3
1.2.3 Association Rule Mining.....	4
1.2.4 Application Area of Data Mining	7
1.3 Problem Statement.....	9
1.4 Aim and Research Objective.....	10
1.5 Research Contributions.....	10
1.5.1 Efficient search space exploration technique	10
1.5.2 Efficient pruning mechanism (PUCP)	11
1.6 Structure of Thesis	11
Chapter 2. Problem Background.....	13
2.1 Introduction	13
2.2 Preliminaries.....	13
2.3 Challenges of HUIM.....	16
2.4 Open Issue	16
Chapter 3. Literature Survey.....	18

3.1 Introduction	18
3.2 Candidate generation and test-based approach (Apriori based)	19
3.3 Tree based approaches	23
3.4 List based approaches	28
3.4.1 HUI-Miner	29
3.4.2 mHUI-Miner	33
3.4.3 ULB-Miner	35
3.4.4 HUI-Miner*	38
3.4.5 UBP-Miner	39
3.5 Research Gap	40
Chapter 4. SCAO based Search Space Exploration Technique for HUIM.....	42
4.1 Introduction	42
4.2 Search Space Exploration in HUIM.....	42
4.3 SCAO-based search space exploration	44
4.3.1 A Construction of Utility list of 2-itemset and k-itemset.....	47
4.3.2 Pruning Mechanism	49
4.3.3 Analysis of Proposed Method Vs. State-of-the-art methods	50
4.4 An illustrative example.	52
4.5 Performance Evaluation.....	53
4.5.1 Experimental Environment.....	53
4.5.2 Performance Evaluation with HUI-Miner.....	54
4.5.3 Performance Evaluation with mHUI-Miner.....	56
4.5.4 Performance Evaluation with ULB-Miner	57
Chapter 5. PUCP-Miner: Predicted Utility Co-exist Pruning	60
5.1 Introduction	60
5.2 The PUCS (Predicted Utility co-exist Structure)	60
5.3 The PUCP (Predicted Utility co-exist Pruning)	61
5.4 Performance Evaluation	65
5.4.1 Execution Time Analysis.	65
5.4.2 Memory Analysis.....	68
Chapter 6. Conclusion and Future Work.....	71
6.1 Conclusion.....	71
6.2 Future Work.....	72
REFERENCES.....	73
List of Publications.....	79

List of Abbreviations

DSS	:	Decision Support System
FIM	:	Frequent Itemset Mining
HUIM	:	High Utility Itemset Mining
DM	:	Data Mining
KDD	:	Knowledge Discovery in Database
CRM	:	Customer Relationship Management
FI	:	Frequent Itemset
PUCS	:	Predicted Utility Co-exist Structure
PUCP	:	Predicted Utility Co-exist Pruning
SCAO	:	Support Count Ascending Order
DB	:	Database
HUI	:	High Utility Itemset
TU	:	Transaction Utility
TWU	:	Transaction Weighted Utility
HUI-Miner	:	High utility Itemset miner
mHUI-Miner	:	Modified High Utility itemset miner
ULB-Miner	:	Utility List Buffer miner
TRid	:	Transaction Identification
iutility	:	itemset's utility
rutility	:	remaining utility
OOA	:	Objective-Oriented Association
TWDC	:	Transaction Weighted Downward Closure
HTWU	:	High Transaction Weighted Utility
IIDS	:	Isolated Item Discarding Strategies
DCG	:	Direct Candidates Generation
FUM	:	Fast Utility Mining
HYP-Tree	:	High-Yield Partition Tree

ARM	:	Association Rule Mining
UPM	:	Utility Pattern Mining
HUC-Prune	:	High-Utility Candidate Prune
UP-Tree	:	Utility Pattern Tree
UP-Growth	:	Utility Pattern growth
DLN	:	Decreasing Local Node utilities
DLU	:	Discarding Local Unpromising items
DGN	:	Decreasing Global Node Utilities
DGU	:	Discarding Global Unpromising items
PHUI	:	Potential High Utility Itemsets
CHUI-Mine	:	Concurrent High Utility Itemset Mining
FHM	:	Fast algorithm for High utility itemset Mining
BEO	:	Bit mErge cOnstruction
PU	:	Predicted Utility
IHUP	:	Incremental High Utility Pattern
IHUPTF	:	Incremental High Utility Pattern Transaction Frequency
IHUPPL	:	Incremental High Utility Pattern Lexicographic
IHUPTWU	:	Incremental High Utility Pattern Transaction Weighted Utility
FP	:	Frequent Pattern
SL	:	Summary List
SUL	:	Summary of Utility list
ShFSM	:	Share counted Frequent Set Mining

List of Figures

Figure 1.1	Classification	3
Figure 1.2	Clustering	3
Figure 1.3	Knowledge Discovery from database	4
Figure 1.4	Data mining application area	8
Figure 3.1	Publication trend of HUIM	18
Figure 3.2	Classification of HUIM approaches	19
Figure 3.3	Set enumeration tree	32
Figure 3.4	Global IHUP Tree	34
Figure 3.5	Local Prefix Tree for Tc	34
Figure 3.6	Local Prefix Tree for Tcd	34
Figure 3.7	Utility list buffer for sample database	36
Figure 3.8	Sum of utility list for sample database	36
Figure 3.9	Utility list buffer after inserting itemset {CD}	37
Figure 3.10	Summary of Utility list after inserting itemset {CD}	37
Figure 3.11	Utility list buffer after inserting itemset {CA}	37
Figure 3.12	Summary of Utility list after inserting itemset {CA}	37
Figure 3.13	Utility list buffer after inserting itemset {CB}	37
Figure 3.14	Summary of Utility list after inserting itemset {CA}	38
Figure 4.1	Set Enumeration Tree	43
Figure 4.2	Proposed Model of SCAO-based search space exploration technique	44
Figure 4.3	Set enumeration tree of a sample dataset	47
Figure 4.4	Number of comparisons (HUI-Miner Vs. SCAO-HUI-Miner)	55
Figure 4.5	Execution time comparison (HUI-Miner Vs. SCAO-HUI-Miner)	55
Figure 4.6	Number of comparisons (mHUI-Miner Vs SCAO-mHUI-Miner)	56
Figure 4.7	Execution time comparison (mHUI-Miner Vs SCAO-mHUI-Miner)	57
Figure 4.8	Number of comparisons(ULB-Miner Vs SCAO-ULB-Miner)	58
Figure 4.9	Execution time comparison (ULB-Miner Vs SCAO-ULB-Miner)	59

Figure 5.1	PUCS Structure	61
Figure 5.2	PUCS of the sample database	61
Figure 5.3	Proposed Model for PUCP-Miner	62
Figure 5.4	Execution time comparison (HUI-Miner Vs PUCP-Miner)	66
Figure 5.5	Execution time comparison (mHUI-Miner Vs PUCP-Miner)	67
Figure 5.6	Execution time comparison (HUI-Miner Vs PUCP-Miner)	68
Figure 5.7	Memory comparison (mHUI-Miner Vs PUCP-Miner)	69
Figure 5.8	Memory comparison (ULB-Miner Vs PUCP-Miner)	70

List of Tables

Table 1.1	Sample database for FIM	5
Table 2.1	Transaction database	14
Table 2.2	Utility table	14
Table 2.3	Sample transaction database	14
Table 2.4	Sample Utility Table	14
Table 2.5	Sample transaction Database	16
Table 2.6	Sample utility Table	16
Table 3.1	Summary of apriori based approaches for HUIM	22
Table 3.2	Summary of apriori based approaches for HUIM	26
Table 3.3	Sample transaction database	29
Table 3.4	Sample utility table	29
Table 3.5	Sample revised database	29
Table 3.6	Initial utility list of 1-itemset	30
Table 3.7	Utility list of 2-itemset	31
Table 3.8	Summary of utility list based approaches for HUIM	39
Table 4.1	Sample Transaction Database	45
Table 4.2	Sample Utility Table	45
Table 4.3	Utility Lists of 1-Itemset	45
Table 4.4	Utility Lists of 2-Itemset	48
Table 4.5	Utility Lists of 3-Itemset	48
Table 4.6	Utility Lists of 4-Itemset	48
Table 4.7	Support count and TWU of item	52
Table 4.8	Revised Transaction dataset	52
Table 4.9	Characteristics of Dataset	54
Table 4.10	No of comparisons (HUI-Miner Vs SCAO-HUI-Miner)	54
Table 4.11	Execution time comparison (HUI-Miner Vs SCAO-HUI-Miner)	55
Table 4.12	No of comparisons (mHUI-Miner Vs SCAO-mHUI-Miner)	56
Table 4.13	Execution time comparison (mHUI-Miner Vs SCAO-mHUI-Miner)	57

Table 4.14	No of comparisons (ULB-Miner Vs SCAO-ULB-Miner)	58
Table 4.15	Execution time comparison (ULB-Miner Vs SCAO-ULB-Miner)	58
Table 5.1	Sample Transaction Database	61
Table 5.2	Sample Utility Table	61
Table 5.3	Dataset characteristics used in experiments	65
Table 5.4	Execution time comparison (HUI-Miner Vs PUCP-Miner)	66
Table 5.5	Execution time comparison (mHUI-Miner Vs PUCP-Miner)	67
Table 5.6	Execution time comparison (ULB-Miner Vs PUCP-Miner)	67
Table 5.7	overall improvement of proposed approaches	68
Table 5.8	Memory Requirement (mHUI-Miner Vs PUCP-Miner)	69
Table 5.9	Memory Requirement (ULB-Miner Vs PUCP-Miner)	70

CHAPTER-1

Introduction

1.1 Introduction

Nowadays, businesses are rapidly growing due to globalization [1]. Any business organization has to design an efficient DSS (Decision Support System) to sustain tough competition [2]. In today's era, every business organization generates a vast volume of data related to products, services, customers, etc. Every year, approximately 15 exabytes of fresh data are produced by various organizations [3]. The majority of data is unprocessed and useless. In the current scenario, data is an organizational asset if appropriately used to design an efficient DSS [4]. The knowledge/information hides inside the raw data that can be utilized to design an efficient DSS. It is practically impossible to derive the knowledge/information from the raw data manually [3] [5]. Data mining comprises a collection of techniques to automatically discover the knowledge hidden inside the raw data [6].

FIM (Frequent Itemset Mining) is a well-known data mining method to find out the vital pattern for designing DSS in business [7]. FIM is also used to study the customer buying pattern from the historical data in the superstore [8]. FIM identifies a set of items that are repeatedly appear together from the transaction database. A common issue in transaction databases is the identification of frequent itemsets, which comprise items regularly purchased by customers in numerous transactions [9]. To illustrate, consider the scenario where many customers buy noodles and spicy sauce together. These discernible patterns hold significant value for human comprehension and can effectively support and inform decision-making processes. For instance, the {noodles, spicy sauce} combination might guide marketing strategies such as promoting noodles and spicy sauce. The exploration of frequent itemsets constitutes an extensively researched task in the area of data mining, with its applications including various domains. Essentially, it entails the analysis of a database to unveil instances where specific values (items) co-occur within a collection of database entries, i.e. (transactions) [10]. An item's frequency is not much interesting for the users as it shows only the number of times the item present in the dataset [11]. In reality, more interest

is highly profitable items, whereas the most valuable customers yield high profits in the retail business. In the medical domain, the symptoms that contribute more to the diseases are most important [12]. The patterns not discovered in FIM include rarely purchased but highly profitable items, customers who purchased rarely but generated high profit [13], or the symptoms that rarely appear but contribute more to the diseases [12]. It is necessary to consider the item's utility in the analysis. The item's utility shows its usefulness like quantity, importance, profit, weight, etc. [14]. To overcome the limitation of FIM, the HUIM (High Utility Itemset Mining) problem has been defined [11]. The HUIM discovers the set of items that generate a high profit for the retail business, identifies a valuable customer that made a high profit for the business, and finds more significant symptoms for the disease[8][15]. Unlike the FIM, HUIM considers the importance and quantity of items, so the pattern generated from HUIM is more important than FIM for designing a DSS.

1.2 Background Study

1.2.1 Data Mining

In the era of globalization and open markets, every business struggles to sustain tough competition [16]. In a recent scenario, huge amounts of data about the product, customer, sales, production, consumption, expenditure, etc., are produced by various organizations [17, 18]. To dominate the market, the organization needs to utilize this data for designing various decision support systems about production, sales, customer relationships etc. [19]. Manual analysis of this huge data is practically impossible. Automatic methods are required to analyze the huge amounts of data. DM (Data Mining) plays a vital role in data analysis to discover useful knowledge. DM provides the tools and techniques to automatically discover the knowledge from the massive raw data [20]. DM is also known as KDD (Knowledge Discovery in Database) [20]. Data mining methods like association rules, classification, clustering, genetic algorithms, neural networks, nearest neighbor methods and artificial intelligence are used to find out the unseen knowledge from the data [20, 21].

- **Classification:** - Assigned objects in a collection to specify classes or categories. The objective is to predict the target class for new cases [22].

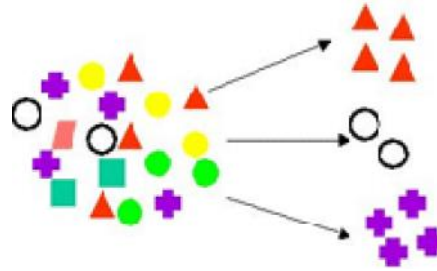


Figure 1.1: Classification

- **Clustering:-** Grouping the objects based on the information found in the data describing the objects or their relationships [23].

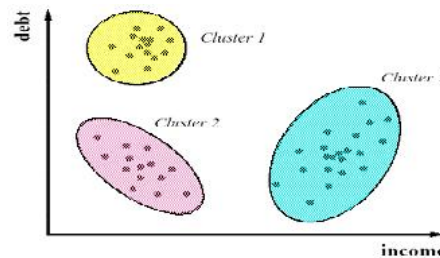


Figure 1.2: Clustering

- **Association Rules:** - Association rule mining finds the relationship between the itemsets and shows how frequently the itemset relates to others [24].

1.2.2 Knowledge Discovery From Database

KDD is the structured method for discovering useful patterns from massive, complicated datasets[25]. The derived pattern from the KDD process is comprehensible, useful, innovative, and valid. The core of the KDD process is data mining[26]. The KDD process follows the below-mentioned steps[25]. The various steps for the KDD process are presented in Figure 1.3.

- **Selection:** It selects the portion of the data from the massive data from which a pattern can be found. The primary objective is to transform the original dataset into a target dataset.
- **Pre-processing:** The collected data may not be directly used for the analysis. It should be cleaned and transformed before using it. This step involves activities like data integration, normalization, noise removal, etc.

- **Transformation:** It includes reducing the data and converting it into an appropriate presentation for the DM techniques.
- **Data mining:** Applied appropriate data mining methods or algorithms to extract the interesting patterns. These methods include clustering, classification, and association rule mining. It discovers the patterns used to design a typical business decision support system.
- **Evaluation:** The discovered patterns and relationships are interpreted in this step. The pattern generated by data mining steps is assessed to see if it is useful or not. Otherwise, the process of the KDD is restarted from the previous step.

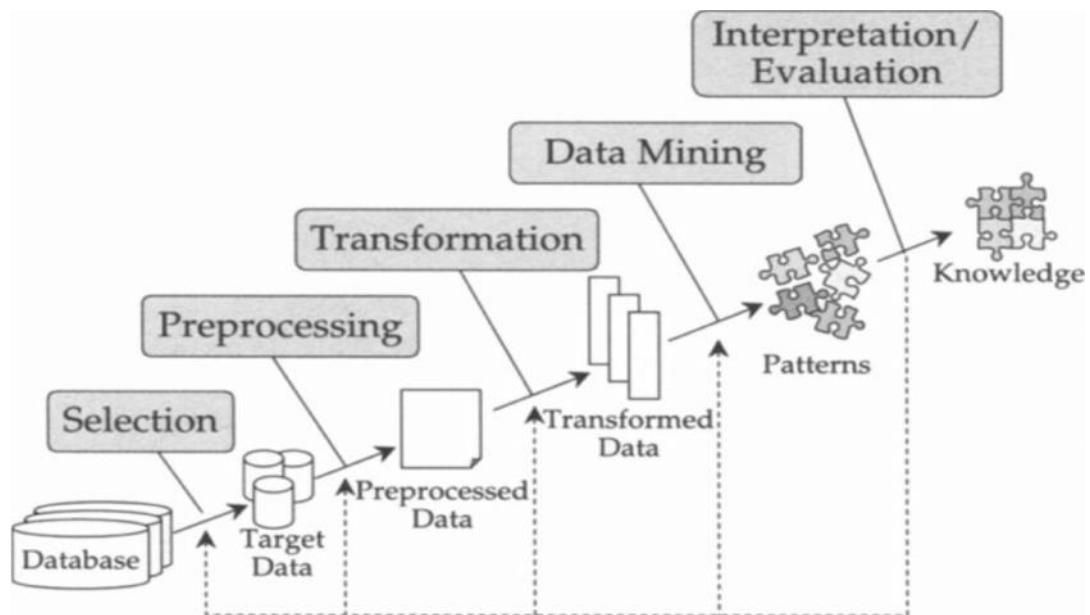


Figure 1.3: Knowledge Discovery from Database

1.2.3 Association Rule Mining

ARM (Association Rule Mining) is used to analyze a large volume of data to uncover intriguing linkages and relationships. This rule displays the number of times an itemset appears in a dataset[27]. The association rule is derived the correlation among the items [7]. It is a two-step procedure. Initially, it determines the frequent pattern from the extensive data, and secondly, it generates interesting association rules[20]. For the given set of items and set of transactions in the retail dataset, the transaction contains the collection of items. The expression of the association rule $A \rightarrow B$ where A and B are the set of items. The rule indicates the transactions that include itemset A tend to have itemset B[2][29].

1.2.3.1 Frequent Itemset Mining

FIM (Frequent Itemset Mining) is the preliminary step for the ARM. In the transaction database, FIM identifies set of items that exist together frequently. The frequency is considered as the quantity of transactions containing itemset. If the frequency of the itemset is more than the user-defined *min_support* threshold, then the itemset is frequent [30].

Definition 1.1: support count ()

The support count of itemset is defined as the number of transactions containing itemset. Consider the itemset X,

$$\sigma(X) = | \{ T_i \mid X \subseteq T_i, T_i \in T \} | \quad (1.1)$$

Where the $|\cdot|$ defines the count of transactions T_i .

Consider the sample database as described in Table 1.1. For the itemset $X = \{\text{Pen, Pencil}\}$, the support count of X is three as pen and pencil appear in three transactions.

Table 1.1: Sample database for FIM

Transaction-ID	Items
1	Pen, Pencil, Eraser,
2	Pencil, Eraser, Sharpener
3	Pen, CD, DVD
4	Eraser, CD
5	Pen, CD, DVD
6	Pen, Pencil, Eraser, Sharpener
7	Pen, Eraser, Sharpener, DVD
8	Eraser, CD, DVD
9	Sharpener, CD, DVD
10	Pen, Pencil, Eraser, Sharpener, CD, DVD

Definition 1.2: Frequent Itemset

*For an itemset X, if the support count of itemset X is not less than the user-defined *min_support* threshold than X is called FI (Frequent Itemset).*

$$\sigma(X) \geq \text{min_support} \quad (1.2)$$

Consider the itemset $X = \{\text{pencil, eraser}\}$ of database mentioned in the Table 1.1 and the user-defined min_support is 35%. Then the itemset X is a frequent itemset as its support count $\sigma(X)$ is four, i.e. 40%, which is not below the min_support threshold.

1.2.3.2 Generation of association rule

The coexistence of the itemsets is described by association rule. Consider frequent itemset $X = \{I_1, I_2, I_3, \dots, I_n\}$. A and B are the two subsets of X , $A \cap B = \emptyset$, $A \neq \emptyset$, $B \neq \emptyset$, and $A \cup B = X$, then the association rule $A \rightarrow B$ holds the correlation between itemset A and B . Support and confidence are the measures to evaluate the strength of the rule[31].

Definition 1.3: Support (s)

Support of the rule indicates the number of transactions containing the group of itemsets.

It decides how frequently the rule is relevant to a given dataset. Consider the support (s) of rule $A \rightarrow B$ that indicates the percentage of the transactions containing $A \cup B$ in a given dataset of N transactions[20].

$$s(A \rightarrow B) = \frac{\sigma(A \cup B)}{N} \quad (1.3)$$

Consider the frequent itemset $X = \{\text{pencil, eraser}\}$ from the dataset in Table 1.1. For the association rule $\text{pencil} \rightarrow \text{eraser}$, the support (s) of the rule is 4/10, i.e. 40% or 0.4.

Definition 1.4: confidence (Conf)

For the rule $A \rightarrow B$, the confidence of rule is $\text{Conf}(A \rightarrow B)$, which indicates that the percentage of times A appears and also appears B

Confidence shows the reliability of the rule. Consider the confidence of rule $A \rightarrow B$ is $\text{Conf}(A \rightarrow B)$, which indicates the percentage of times A appears and also appears B . The conditional probability is $P(B|A)$ [20].

$$\text{Conf}(A \rightarrow B) = P(B|A) = \frac{\sigma(A \cup B)}{\sigma(A)} \quad (1.4)$$

The strong association rule satisfies the user-defined (min_support) minimum support and (min_conf) minimum confidence. Consider the association rule $\text{pencil} \rightarrow \text{eraser}$ derived from

the database shown in Table 1.1. The confidence Conf (pencil \rightarrow eraser) is one as 100 percent of the time the transaction contains pencil also contains eraser.

Association rule mining is the familiar data mining technique to discover useful patterns from a large amount of transactional data of various domains like retail stores, medical, etc.

FIM is the subfield and foundation of association rule mining. The main limitation of the FIM is that it considers the presence or absence of the item. The rules derived from the FIM can guide only the co-existence of the items. FIM identifies the set of item that appear frequently together in the transaction database. It does not consider the item's importance/weight and the quantities. In real-world applications, it is necessary to consider the item's quantities and importance (utility) to design an efficient DSS (Decision Support System). The HUIM (High Utility Itemset Mining) problem has been defined to address this FIM issue[32][33]. Unlike FIM[7][34], the item's quantities and importance are considered in the HUIM problem. The association rules derived from the HUIM are more significant than the FIM in designing a decision support system. HUIM finds the set of items that yield a high profit from the transaction database, identifies the valuable customer for the business to contribute to high profit [35], and discovers the symptoms that contribute more to the disease.

1.2.4 Application Area of Data Mining

Data mining serves as a potent tool employed across diverse industries for extracting valuable insights and patterns from extensive datasets. These insights can be strategically leveraged to gain a competitive edge within the knowledge-based economy. Figure 1.4 represents various application area of data mining[36][37].

- ❑ **CRM (Customer Relationship Management):** Data mining empowers businesses to scrutinize customer data, uncovering patterns and preferences. This information can be harnessed to personalize marketing initiatives, enhance customer service, and bolster customer retention efforts.

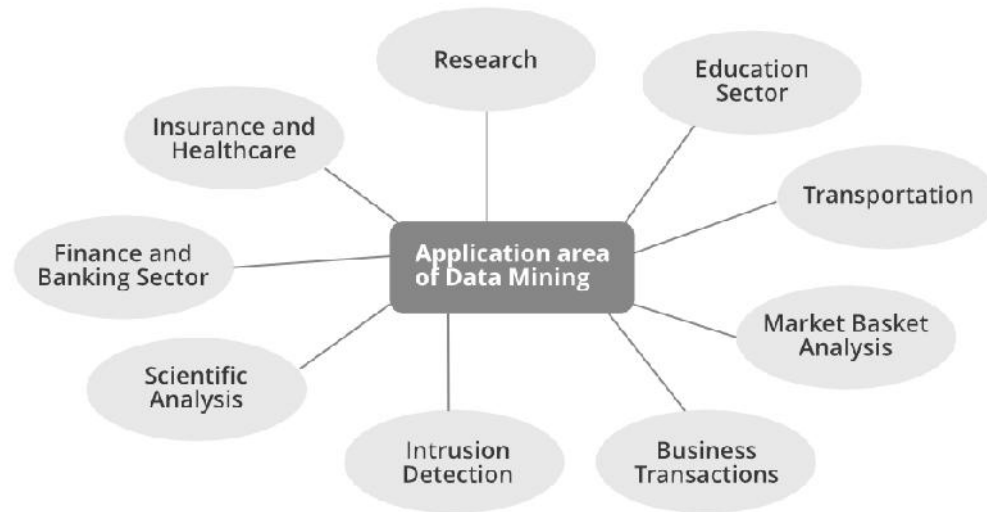


Figure 1.4: Data mining application area

- ❑ **Market Basket Analysis:** Retail establishments utilize data mining to scrutinize customer purchase histories, revealing associations between the products. This, in turn, allows for strategic product placement and promotional optimization, ultimately bolstering sales.
- ❑ **Fraud Detection and Prevention:** Banking and financial institutions utilize data mining to identify fraudulent activities by meticulously analyzing transaction data and searching for unusual patterns or anomalies that may indicate fraud.
- ❑ **Healthcare and Medical Research:** Data mining is instrumental in dissecting electronic health records, clinical data, and genomic information. This analytical process uncovers disease patterns, refines treatment strategies, and ultimately enhances patient outcomes.
- ❑ **Recommendation Systems:** Corporations like Netflix and Amazon harness data mining techniques to propose products or content tailored to individual users considering their historical preferences and behaviors.
- ❑ **Manufacturing and Quality Control:** DM can be employed in manufacturing processes to pinpoint defects, optimize production procedures, and minimize downtime by predicting equipment failures.
- ❑ **Sentiment Analysis:** Social media platforms and businesses employ data mining to assess text data and user comments, gauge public sentiment, track brand perception, and inform strategic marketing decisions.

- ❑ **Supply Chain Management:** Data mining plays a pivotal role in streamlining supply chain operations. Analyzing historical data facilitates demand forecasting, reduces inventory expenses, and improves logistics.
- ❑ **Energy Consumption Analysis:** Utility providers employ data mining to scrutinize energy consumption patterns. This analysis guides distribution optimization, peak load management, and the implementation of energy-saving initiatives.
- ❑ **E-commerce and Pricing Optimization:** E-commerce platforms employ DM to dynamically adjust pricing considering the factors such as demand, competitive pricing, and customer behavior, thus maximizing profits.
- ❑ **Human Resources:** Data mining aids in talent acquisition, employee retention, and workforce planning by scrutinizing employee data to identify high-performing candidates.
- ❑ **Environmental Monitoring:** Data mining is used in analyzing environmental data encompassing weather patterns, pollution levels, and ecological trends. It aids in conservation efforts and facilitates disaster prediction.
- ❑ **Education:** Educational institutions employ data mining to assess student performance data, allowing the identification of at-risk students, personalization of learning experiences, and enhancements in educational outcomes.
- ❑ **Telecommunications:** DM is used to scrutinize call data records, enabling fraud detection, optimizing network performance, and predicting customer churn in telecom enterprises.
- ❑ **Transportation and Logistics:** Data mining is pivotal in optimizing routes, predicting vehicle maintenance requirements, and forecasting demand for transportation services.

1.3 Problem Statement

According to the studies by C.Zhang et al. (2018)[15], P F. Viger (2019)[38], and W.Gan et al. (2019)[39], utility list-based HUIM approaches are recent and better performing than other two-phase and pattern growth approaches. However, the performance of the utility list-based approaches is limited due to several costly utility list join operations. The focused problem statement of this research is:

“To propose efficient high utility itemset mining approach which reduces the cost of the utility list join operations and minimize the join operations as well.”

1.4 Aim and Research Objective

The aim of the research is to develop an efficient approach for HUIM by eliminating the unnecessary utility list join operations and reducing comparisons required to search a common elements between two utility lists in join operation. The proposed research work is the following objectives:

- To study and investigate existing methods for the HUIM.
- To identify the challenges for the HUIM.
- To identify the scope to enhance the HUIM algorithm's performance.
- To develop and investigate the efficient method for exploring the search space to reduce the cost of utility list join operations by minimize the comparisons required to join utility lists.
- To design a novel structure to store the itemset's predicted utility which is used to develop an effective pruning mechanism.
- To develop and investigate an efficient pruning mechanism to decrease the join count by eliminating unnecessary join operations of utility lists.
- To implement and assess the effectiveness of the suggested methods.
- To compare the results of the proposed approaches with existing state-of-the-art methods.

1.5 Research Contributions

The main contribution of the research is to design efficient utility list join operations by decreasing the comparisons. The comparisons required to find out the common elements between the utility lists can be decrease by efficiently exploring the search space. To design an effective pruning mechanism, PUCP (Predicted Utility Co-exist Pruning) to decrease the number of join operations.

1.5.1 Efficient search space exploration technique

The contribution of the research is to design efficient utility list join operations by reducing the comparisons required to find the common elements between the utility lists. Comparisons can be reduces by efficiently exploring the search space. The proposed SCAO (Support Count Ascending Order) based search space exploration technique reduces the cost

of utility list join operations by decreasing comparisons required to find common transactions between utility lists.

1.5.2 Efficient pruning mechanism (PUCP)

The significant contribution of the research is study the items co-existence in the dataset. Predicted Utility Co-exist Structure, known as PUCS, proposed to store the utility data. Predicted Utility Co-exist Pruning, known as PUCP, proposed eliminating unnecessary utility list join operations. PUCP mechanism greatly reduces the utility list join operations and thus improves the algorithm's performance. It eliminates the low utility itemsets directly without performing the join operations.

1.6 Structure of Thesis

Chapter 1 includes the general introductions of the research work. It explains the data mining, the KDD Process, and the various data mining techniques, specifically association rule mining. It also discusses the data mining applications in the real world. The problem statement and objective of the research is also included in this chapter. The chapter ends with the research contribution.

Chapter 2 presents a problem background with defined various terms associated with HUIM. The challenges of the HUIM problem are explained. Also, discuss the open issue for the HUIM at last.

Chapter 3 presents the comprehensive literature survey of HUIM. It shows the various categories of the HUIM approaches. The detailed review about different state-of-the-art algorithms falls under each category described in this chapter. The literature review incorporates existing apriori-based, tree-based, and utility list-based approaches used for HUIM. A summary of reviews and a discussion are also covered. The research gap is explained at the last.

Chapter 4 discussed the efficient SCAO (Support Count Ascending Order) based search space exploration technique. Through the time complexity comparisons, it shows the SCAO based search space exploration technique required less comparisons compare to TWU ascending order based technique used by other state-of-the-art approaches. The chapter is

completed with the experimental result analysis of proposed approach with other state-of-the-art methods on some standard real datasets.

Chapter 5 discusses the proposed PUCP (Predicted Utility Co-exist Pruning) uses the PUCS (Predicted Utility Co-exist Structure). Also discussed the how the PUCP-Miner eliminates the unnecessary utility list join operations. It presents a detailed result analysis and performance comparison with the traditional algorithms on some real datasets. At last performance of the combine approaches SCAO based and PUCP is compared with other state-of-art methods.

Chapter 6 includes a conclusion that summarizes the research findings. It discusses the experimental study's major findings and observations. It also represent the future enhancement of the work.

CHAPTER-2

Problem Background

2.1 Introduction

From the transaction dataset, HUIM (High Utility Itemset Mining) finds the itemsets that yield high profits. HUIM considers the item's quantity in every transaction and the utility of each item in the database.

2.2 Preliminaries

Let set $I = \{i_1, i_2, i_3, \dots, i_n\}$ is the single itemset. Database (DB) is the collection of transactions and utility table is presented in Table 2.1 and Table 2.2. The utility of an individual item is shown in utility table. The transaction consists itemset with quantities. The collection of transactions in the DB are $(T_1, T_2, T_3, \dots, T_m)$. Each transaction is uniquely identified by T_i . The transaction T_i is definite as

$$T_i = \{i_k: q(i_k) | 1 \leq i \leq m, 1 \leq k \leq n\} \quad (2.1)$$

The items in the transaction T_i is a subset of I . The items in T_i are associated with quantities defined as count $q(i_k)$ (where $1 \leq k \leq n$) called an internal utility of the item in the transaction. The utility table maintains the utility (importance/weight/profit) values of each item i in I , known as an external utility. The sample database and utility table are demonstrated in Table 2.3 and Table 2.4, respectively.

Definition 2.1: Item's utility

For the item i_k in the transaction T_j , the item's utility of i_k is $u(i_k, T_j) = q(i_k, T_j) \times p(i_k)$.

i.e. $u(k, T_3) = q(k, T_3) \times p(k) = 15$

Table 2.1: Transaction database

id	Transactions
T1	q (i ₁), q(i ₂), q(i ₃)..... q(i _n)
T2	q (i ₁), q(i ₂), q(i ₃)..... q(i _n)
T3	q (i ₁), q(i ₂), q(i ₃)..... q(i _n)
.
.
.	
Tm	q (i ₁), q(i ₂), q(i ₃)..... q(i _n)

Table 2.2: Utility table

Item	i ₁	i ₂	i _n
Profit	p(i ₁)	p(i ₂)	p(i _n)

Table 2.3: Sample transaction database

Transaction	k	l	m	n	o	p	q
T ₁	-	1	2	-	1	-	-
T ₂	-	-	-	-	-	3	1
T ₃	3	4	-	-	2	-	1
T ₄	1	1	-	1	-	-	-
T ₅	1	2	3	4	5	-	-

Table 2.4: Sample Utility Table

Item	k	l	m	n	o	p	q
Profit	5	1	3	4	2	1	2

Definition 2.2: Itemset's utility

For the itemset P_x in the transaction T_j , the itemset P_x 's utility is

$$(P_x, T_j) = \sum_{ik \in x \quad ik \in Ti} u(ik, Ti) \quad (2.2)$$

i.e. consider itemset $P_x = \{m, n\}$, $u(P_x, T_5) = u(m, T_5) + u(n, T_5) = 25$

Definition 2.3: Itemset's utility in database DB

For the itemset P_x and the transaction database DB, the P_x 's utility in DB is

$$u(P_x) = \sum_{ik \in I \wedge x \in Ti} u(x, Ti) \quad (2.3)$$

i.e. consider itemset $P_x = \{k, l\}$, the itemset P_x exist in transactions T_3 , T_4 and T_5 . So,
 $u(P_x) = u(P_x, T_3) + u(P_x, T_4) + u(P_x, T_5) = 32$

Definition 2.4: Minimum utility threshold

It is the user-specified utility threshold denoted as *MinUtility*.

Definition 2.5: High utility itemset

The itemset P_x is a high utility itemset, if its utility value is higher than the user-defined $MinUtility$ threshold

$$HUI_{set} = \{P_x / P_x \subseteq I, u(P_x) \geq MinUtility\} \quad (2.4)$$

i.e., consider $MinUtility=15$ and itemset $P_x = \{k,l\}$, $u(k,l)$ is 32, which is greater than $MinUtility$, indicating that P_x is a high utility itemset.

Definition 2.6: HUIM Problem.

Extracting all the itemsets from the database which has utility value not lower than the user-specified minimum utility ($MinUtility$) threshold.

Definition 2.7: Support count

The support count of itemset P_x is the transaction count in which itemset P_x exists and is defined as $support(P_x) = |P_x|$

i.e., the support count of itemset $P_x = \{l\}$ is four as itemset l exists in 4 transactions. While the support count of itemset $P_x = \{lm\}$ is 2.

Definition 2.8: Transaction's utility

The transaction's utility is obtained by summing up the utility values of all the items present in that particular transaction. It is defined as

$$TU(T_i) = \sum_{\forall ik \in T_i} u(ik, T_i) \quad (2.5)$$

i.e The transaction utility of transaction T_1 is $TU(T_1) = u(l, T_1) + u(m, T_1) + u(o, T_1) = 9$

Definition 2.9: TWU (Transaction Weighted Utility)

The TWU of an itemset P_x is calculated as the total of the transaction utilities of all transactions containing the itemset P_x .

$$TWU(P_x) = \sum_{P_x \in T_i} TU(T_i) \quad (2.6)$$

i.e. $TWU(n)$ is the total of transaction utility of transaction T_4 and T_5 as n belongs to transactions T_4 and T_5 , So $TWU(n) = TU(T_4) + TU(T_5) = 52$

2.3 Challenges of HUIM

The HUIM problem is much tougher than the FIM problem. FIM, the downward-closure-property states that the frequency (support) of the itemset is anti-monotonic[7], which means supersets of an infrequent itemset are infrequent, and subsets of a frequent itemset are frequent, which is called the Apriori property. It is effective to trim (prune) the search space. But in HUIM, the utility measure is neither anti-monotonic nor monotonic. An itemset with high utility may have a subset or superset with lesser, equal, or more utility[40]. Therefore, the techniques employed in FIM to narrow down the search space, leveraging the downward-closure property of the support, cannot be directly employed in HUIM for search space reduction.

Table 2.5: Sample Transaction Database

Transaction	A	B	C	D
T1	3	0	2	4
T2	0	4	1	0
T3	4	1	3	1
T4	1	1	0	1
T5	0	6	2	0

Table 2.6: Sample Utility Table

Item	A	B	C	D
Profit	5	7	2	1

Consider user-specific threshold, $MinUtility = 45$ for the sample database mentioned in Tables 2.5 and 2.6. The utility of itemset $X = \{A, C, D\}$ is 50, which is more than the $MinUtility$. So, X is the high utility itemset. Another itemset $Y = \{C\}$. Utility of Y is 18 less than $MinUtility$. So, Y is not a high utility itemset even if Y is a subset of X . Now consider itemset $P = \{A, B, D\}$. Utility of P is 41 $MinUtility$, and a subset of P is $Q = \{B\}$, the utility of Q is 84 $MinUtility$. So, P is not a high utility itemset even if it is superset of Q . where itemset Q is the high utility itemset. So pruning the search space is the challenging task in HUIM problem.

2.4 Open Issue

Pruning the search space is a complex and expensive task in HUIM because the utility measure is neither anti-monotonic nor monotonic. The existing methods generate more

candidate itemsets, uses inefficient pruning measures. So the current methods for HUIM suffer from time-consuming operations and high memory requirements. These approaches employ multiple pruning measures that tend to overestimate, resulting in unnecessary computations and increased resource usage. Utility list-based HUIM approaches are recent and outperforming as they do not generate candidate itemsets. However, the performance of list-based algorithms is limited by the need to perform many expensive utility list join operations. These join operations contribute to the computational overhead and can affect the efficiency of the algorithms. Therefore, there is a need to address the computational cost associated with utility list join operations to improve the performance of list-based HUIM algorithms. The cost of join operations in utility list-based approaches is directly proportional to the comparisons needed to find common transactions between the utility lists. Join count/s and number of comparisons are the challenges to decreasing the cost of the utility list join operations. Hence, reducing join count/s and number of comparisons will enhance the performance of the mining algorithm in execution time and memory consumption.

CHAPTER-3

Literature Survey

3.1 Introduction

Identifying high utility itemsets in transaction datasets is a formidable task, even with continuous advancements over the past two decades. Figure 3.1 depicts the publication trends of different approaches for HUIM from 2001 to 2022. The publication data is collected from Google Scholar, using the search query 'high utility itemset mining'. From the statistics, it has been observed that HUIM is the latest topic in the research community, as discovering the utility-oriented pattern is cost consuming task.

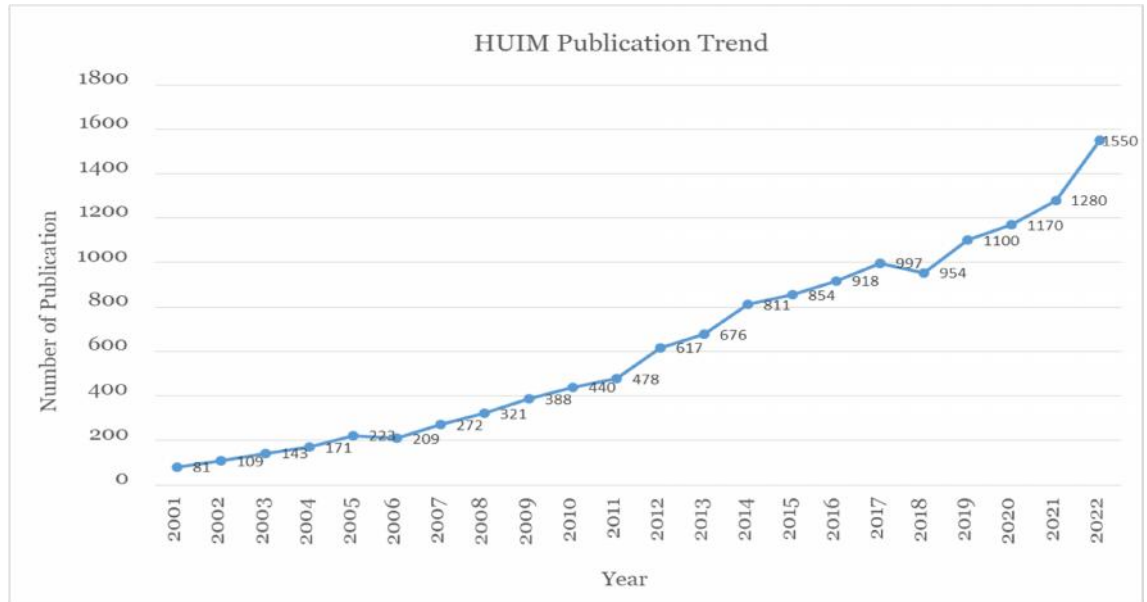


Figure 3.1: Publication trend of HUIM

Several Approaches for HUIM have been suggested using various mining ideologies, data structures, search space exploration, and pruning techniques. Existing HUIM algorithms are broadly classified into three categories based on various mining techniques and data structures.

- 1) Candidate generation and test-based approaches (Apriori-based)
- 2) Tree-based approaches
- 3) Utility list-based approaches

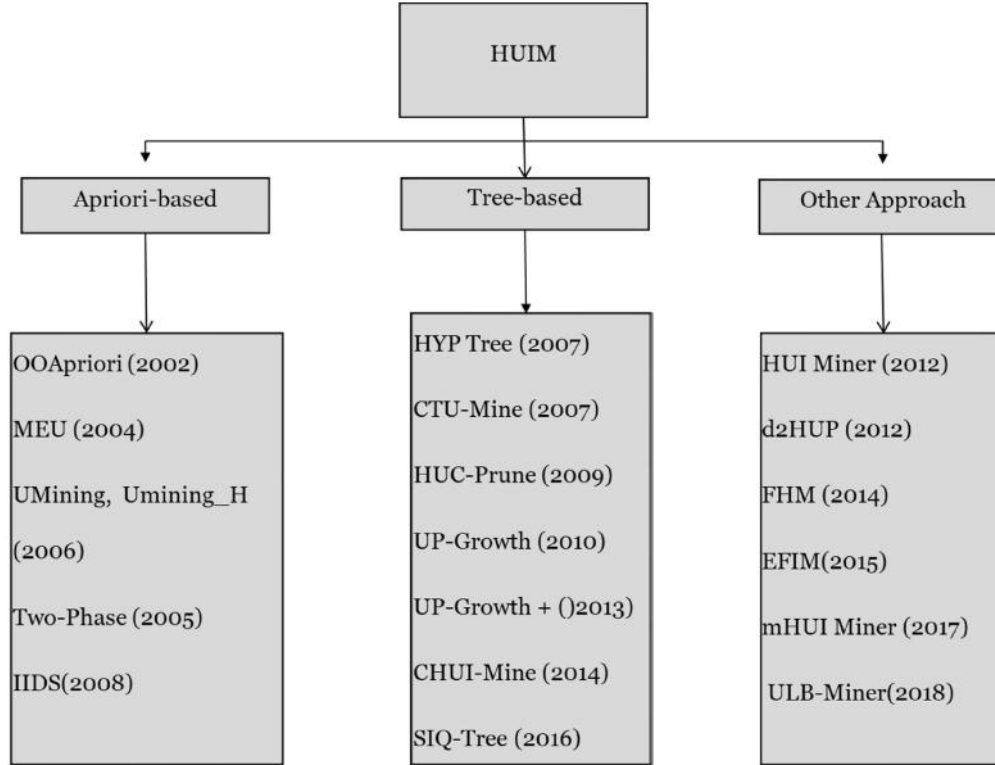


Figure 3.2: Classification of HUIM approaches

3.2 Candidate generation and test-based approach (Apriori based)

In 1994, Shrikant and Agrawal introduced the widely acknowledged downward closure principle, called the apriori property[7]. It states that any superset of non-frequent itemset is non-frequent and any subset of frequent itemset is also frequent. It is very effective to reduce the search space. Candidate sets generation is the core of these apriori-based HUIM approaches. It performs join and prune operations. In the join operation, it generates the k -itemsets from the high utility $(k-1)$ -itemsets. In the prune operation, it discards some of the low utility itemsets based on the utility upper bound, hence reducing the size of candidate sets.

In the year 2002, Shen, Zhang, and Yang introduced an innovative approach to utility mining, explicitly focusing on user-specified objectives[41]. Authors incorporate utility

constraints into traditional association rule mining framework. This approach aims to model association patterns that are in accordance with user-defined objectives and utility requirements. In this methodology, a collection of itemsets is generated using the established Apriori algorithm, ensuring that they satisfy the minimum support requirement. Following this, a pruning process is applied to the association rules, guided by user-defined constraints and utility thresholds. Any rule that fails to align with the user's objectives or does not meet the minimum utility threshold are systematically removed. The utility constraint for OOA (Objective-Oriented Association) rules does not exhibit a monotone or anti-monotone behavior. So, it is important to highlight that the OOA rules do not adhere to a monotonic or anti-monotonic pattern, which can result in removing certain high utility items. Chan et al. initially introduced the utility mining concept and put forward an objective-driven mining algorithm for discovering the top-k closed-utility patterns in the year 2003[42].

In 2004, Y. D. Shen, Z. Zhang, and Q. Yang introduced a theoretical model for high utility itemset[11]. This model defined two key properties: utility bound and support bound, which were aimed at sinking the search space. To complement this, a theoretical framework, a heuristic-based model, has been designed to estimate the utility of k-itemsets based on the utility values of (k-1)-itemsets. While the authors initially proposed a theoretical model and provided theoretical proof, it is imperative to validate its practical applicability on real datasets. The pruning technique employed in this model relies on the utility bound and item support bound, which tend to overestimate. It fails to preserve the downward closure properties. Consequently, the derived results from the estimations may be incomplete.

In 2005, Yao and Hamilton introduced a pair of algorithms for HUIM, denoted as UMining and UMining_H[43]. Both algorithms follow a level-wise approach for generating and testing candidate itemsets. They extensively delve into various mathematical properties to devise effective pruning methods aimed at reducing the candidate itemsets for the subsequent levels. UMining incorporates innovative pruning techniques based on the utility upper bound property. This property implies that the utility value of a k-itemset is bounded by the utility values of all its subsets with a size of k-1.

In contrast, UMining_H relies on heuristic pruning strategies grounded in the expected utility upper bound property. While UMining ensures the discovery of all high utility itemsets, UMining_H overlooks certain high utility itemsets due to its reliance on expected

utility, potentially leading to the removal of some valuable itemsets. These two algorithms were tested using a database containing 8 million records and 2238 unique items. It's important to note that these pruning strategies may not be entirely efficient, as they do not adhere to the downward closure property.

In 2005, Liu, Liao, and Chaudhary addressed the challenge presented by utility measures that do not adhere to the downward closure property by introducing a Two-Phase algorithm[44]. This algorithm employs pruning techniques based on TWU (Transaction Weighted Utility), which demonstrates anti-monotonic behavior. Specifically, it establishes that an itemset's utility falls below the *MinUtility* threshold if the transaction-weighted utility of its subset is below this threshold. This approach proves to be an effective way to reduce the search space.

During the initial phase of the algorithm, it identifies candidate itemsets with transaction-weighted utility equal to or exceeding the *MinUtility* threshold. It prunes the search space using the TWDC (Transaction Weighted Downward Closure) property. It begins by discovering all high transaction-weighted utility 1-itemsets and progressively generates k-candidate HTWU (High Transaction Weighted Utility) itemsets from the (k-1) candidate itemsets. The algorithm evaluates the TWU value of these candidate itemsets by scanning the database and performing pruning at each iteration.

In the second phase, the algorithm calculates the actual utility of each candidate itemset and identifies the high utility itemsets. However, a notable limitation of the Two-Phase algorithm is that it generates candidate k-itemsets by combining (k-1)-itemsets, potentially introducing patterns that do not actually exist in the database. Consequently, a significant amount of processing time is required for candidate itemsets, even the itemsets not exist in the database. Additionally, for calculating the TWU in each iteration, the algorithm frequently scans the entire database. Furthermore, it stores all candidate itemsets in memory before executing the second phase. Since TWU is not a tight bound of utility measure, the Two-Phase algorithm generates a more candidate sets, even when only a few high utility itemsets are present in the database.

The problem of itemset sharing mining[45] can be transformed into a utility mining problem by substituting the item's frequency with its profit value. In 2008, Li, Yeh, and Chang introduced strategies[46] for removing isolated items to diminish the candidate set size during the mining process at each level. With each pass, these IIDS (Isolated Item Discarding

Strategies) involve scanning a dataset that is smaller than the original one by omitting the isolated items. This approach entails scanning the database at each level while progressively reducing its size, ultimately enhancing the efficiency of the multiple-pass, level-wise candidate generation method. DCG (Direct Candidates Generation)+ [46] and FUM (Fast Utility Mining) were further developed by applying IIDS strategies to DCG and FhFSM[45]. The performance of the DCG+ and FUM is better than the two-Phase, UMining, UMining_H, and MEU. Nevertheless, both methods still face the challenge of generating and evaluating candidates in a level-wise manner, necessitating multiple scans of the database.

Table 3.1: Summary of apriori based approaches for HUIM

Sr. No	Algorithm Name	Key points	Pros	Cons
1	OOApriori	Enhances the association rule mining considering the user objectives and utility	It considers the user objectives so it generates more useful patterns	Pruning using the OOA rules that are neither monotonic nor anti monotonic may miss useful patterns.
2	MEU	The definition of high utility itemset mining was proposed first time. It designs the model to calculate the expected utility using item support bound and utility upper bound.	It considers both purchase quantities and unit profit of the item.	It cannot maintain downward closure property. Complete results are not derived. The proposed model is only theoretical.
3	UMining and UMining_H	uses several mathematical properties of utility measure to prune the search space	UMining uses the utility upper bound and UMining_H utilizes a heuristic pruning strategy	It generates a large amount of candidate Patterns and suffers from excessive candidate generations. Introducing poor scalability

4	Two-Phase	Proposes TWDC property to discover HUIs in two phases.	It solves the problem of complex pruning methods required for HUIM problem as utility measure do not support downward closure property. It can significantly prune huge candidate patterns.	It uses a TWU for pruning, which is also an overestimation. It requires level-wise candidate generation and testing. It also scans the dataset multiple times.
5	IIDS	It identifies and discards the isolated items that will not be a part of HUI.	In level-wise candidate generation, it gradually reduces the database by applying an item discarding strategy to improve algorithm's performance.	It requires level wise candidate generation and testing with multiple database scans.

▪ Summary of Candidate generation and test-based approaches (Apriori based)

Apriori-based approaches generate a candidate set level-wise and prune itemsets based on various measures like TWU, Utility upper bound, expected utility, etc. First, they create a 1-itemset and test if it is high utility itemset or not. Also, perform pruning based on various measures. Then, it generates the candidate sets of 2-itemset by combining the 1-itemsets. Again, it performs the pruning and discovers the high utility itemsets. Recursively, it generates the candidate set of k-itemsets by combining the (k-1) itemsets. Performs pruning and discovers the high utility k-itemset. The main drawback of these approaches is they scan the database multiple times. It generates a huge amount of candidate itemsets as it relies on the loose upper bound for pruning. It also generates some patterns that are not present in the database, so it wastes time for processing these patterns. Large memory is needed to maintain the vast number of candidate itemsets, making the algorithm inefficient.

3.3 Tree based approaches

The apriori approach has significant drawbacks despite its effectiveness in HUIM. These limitations encompass generating an excessive number of candidates, the requirement for

repetitive database scans, and a relatively slow execution speed. To overcome these constraints, researchers have formulated tree-based HUIM algorithms. These tree-based algorithms typically encompass three fundamental stages:

- ✓ **Tree Construction:** In this initial phase, a specialized tree structure is created.
- ✓ **Generation of Candidate HUIs:** Algorithms are applied to generate a set of candidate HUI (High Utility Itemsets) from the tree.
- ✓ **HUI Identification:** From the pool of generated candidates, the genuine high utility itemsets are discerned.

These tree-based HUIM algorithms aim to rectify the limitations associated with the apriori method, offering more efficient and effective approaches for mining high utility itemsets.

Hu et al. in 2007, introduced an approximation technique designed to fix the effect of objective function, predefined utility, and performance metrics, leveraging item attributes[47]. This method identifies high-utility combinations and subsequently identifies HUIs by employing a HYP (High-Yield Partition) tree. Unlike the traditional techniques used in ARM (Association Rule Mining), this approach is specifically designed for pinpointing particular data segments and combinations of items/rules that adhere to specific conditions for maximizing a previously defined objective function. Unlike to earlier UPM (Utility Pattern Mining) methods, it places a significant emphasis on "rule discovery" in relation to individual attributes and, the overall criteria guiding to discover the results. Its core objective is to mine the sets of patterns and rules/items combinations that yield the most substantial contribution to achieve a previously defined objective function[47].

Ahmed et al. introduced an innovative tree-based algorithm named HUC-Prune (High-Utility Candidate Prune)[48]. The suggested HUC tree is a prefix tree designed to hold candidate items sorted in TWU descending order. Each node within the HUC tree includes the item's name along with its corresponding TWU value. HUC-Prune replaces the traditional level-wise candidate generation process with a pattern-growth mining approach, same as the IHUP (Incremental High Utility Pattern) algorithm[49]. This approach scans the database a maximum of three times for find out the HUIs and demonstrates superior performance compared to apriori-based algorithms.

The IHUP algorithm[49], equipped with three distinct tree structures, namely IHUPTF-tree, IHUPPL-tree, and IHUPTWU-tree, was introduced to facilitate the mining of high-utility patterns in an incremental and interactive fashion. The node of the IHUP Tree represent the itemset. The node includes basic information of the itemset, like name, support count, and TWU. This algorithm works in three phases 1) building the IHUP-Tree 2) generating HTWUIs, and 3) discovering the HUIs. The process of creating the IHUP-Tree is the same as the building of FP-Tree[50]. For constructing the IHUP-Tree, the set of HTWUI₁ is arranged in TWU descending order in each transaction, and then the transaction is added to the IHUP-Tree. Then, from the IHUP-Tree, all the HTWUI_k are derived according to the FP-Growth approach in step 2. In the final step from HTWUI_k, all the HUIs are identified by rereading the database. Thus, IHUP avoids the candidate itemset generation. Even though the IHUP has a better performance compared to IIDS and Two-Phase, it produces a huge number of HTWUI in step 1. It also uses the overestimated utility TWU, limiting the IHUP's performance in terms of execution time and memory consumption.

In 2010, Tseng et al. were inspired by the frequency-based FP-Growth algorithm. Authors proposed UP-Tree (Utility Pattern Tree), a compressed tree structure, and a familiar algorithm called UP-Growth (Utility Pattern growth)[51] to discover all HUIs efficiently. UP-Growth incorporates four novel strategies, namely (1) DLN (Decreasing Local Node utilities), (2) DLU (Discarding Local Unpromising items), (3) DGN (Decreasing Global Node utilities during the construction of a global UP-tree), and (4) DGU (Discarding Global Unpromising items during the construction of a global UP-tree). The UP-Tree is generated after scanning the database twice. The transaction utility of each transaction and TWU of individual items are calculated. The unpromising items whose TWU is not satisfied the *MinUtility* threshold are eliminated. Then, the remaining promising items are arranged in TWU descending order in the transactions. The database is scanned again to add the transactions in UP-Tree according to the DGN and DGU strategies. According to DLU and DLN strategies, the PHUIs (Potential High Utility Itemsets) are generated from UP-Tree. In brief, the UP-Growth framework comprises three main steps:

- (1) Scan the database two times to build a global UP-tree, employing the DLU and DLN strategies.
- (2) Iteratively produce PHUIs (Potential High Utility Itemsets) from both the global UP-tree and local UP-trees using UP-Growth and applying the DGU and DGN strategies.
- (3) Determine the ultimate HUIs from the pool of PHUIs. UP-Growth uses the minimum item utility upper bound to generate PHUIs, which is an overestimation.

In 2013, an enhanced version of the UP-Growth, namely UP-Growth+[52] is proposed by Tseng et al. To reduce the overestimation, it uses minimal node utility in each path of the UP-Tree. Compared to UP-Growth, the UP-Growth+ minimizes the number of PHUIs.

Later, in 2013, Song et al. introduced a CHUI-Mine (Concurrent High Utility Itemset Mining) algorithm to find out HUIs by pruning the CHUI-Tree dynamically. For capturing the utility information of the candidate itemset, the author introduced a tree structure called CHUI-Tree. The tree construction process uses dynamic pruning on CHUI-Tree to record the changes in the support count of candidate itemsets. CHUI-Mine employs a concurrent approach, allowing the simultaneous generation of a CHUI tree and the mining of HUIs. As a result, it mitigates the challenge of extensive memory usage typically associated with tree construction and traversal in tree-based HUIM (High Utility Itemset Mining) algorithms. Thorough experimentation has proven that CHUI-Mine surpasses Two-Phase[44] and HUC-Prune[48] not only in efficiency but also in scalability.

Previous tree-based HUIM algorithms suffered from time-consuming tree construction, including multiple scanning of the database, and overestimation utility measure generates a huge number of PHUIs. To deal with these issues in 2016, H Ryang et al. developed a novel tree structure called SIQ-Tree (Sum of Item Quantities) to store database information in a single scan[53]. This efficient approach constructs the initial SIQ tree by scanning the database once. The author also proposed a restructuring method using two novel strategies to reduce the overestimation through which a lower number of candidate patterns are generated. Hence, it improves the mining performance.

▪ Summary of Tree-based approaches

The comparative summary of tree-based HUIM approaches is represented in Table 3.2

Table 3.2: Summary of tree-based approaches for HUIM

Sr. No	Algorithm Name	Key points	Pros	Cons
1	HYP-Tree	An approximation method discerns the contribution of utility.	It is capable of identifying data segments using combinations of a small number of items or rules.	Generate huge HYP tree, and it consumes more memory

2	HUC-Prune	The HUC-tree structure is a prefix tree which stores the items in TWU descending order	It replaces the level-wise candidate generation process by a pattern-growth mining approach.	The upper bound is high, and the generated tree is huge.
3	UP-Growth	Pattern growth algorithm uses the DGU, DGN, DLU, DLN to decrease the overestimation utility	UP-tree is more compact than IHUP-tree, and the strategy is powerful to reduce the number of candidates.	It is time-consuming to process all conditional prefix trees to generate candidates recursively.
4	UP-Growth+	An improved version of UP-Growth which employ two pruning strategies.	The enhanced UP-Growth+ can decrease the overestimated utilities of PHUIs and significantly decreases the candidates.	It is time-consuming to recursively process all conditional prefix trees.
5	CHUI-Mine	Concurrent High-Utility Itemset Mine (CHUI-Mine) algorithm by dynamically pruning the CHUI-tree	It reduces the storage space by avoiding the generation of whole tree. It dynamically prunes the subtree and concurrently mines the HUI	Recursively process all conditional prefix trees is time-consuming.
6	SIQ Tree	Maintain the sum of the item's quantity on the tree. Scan the database only once.	Constructs the SIQ tree with a single scan only and decreases the number of candidate patterns.	It process all conditional prefix trees recursively so, it is time-consuming.

Typically, the tree-based algorithms use different tree construction methods and pattern growth approaches to mine the HUIs in three steps.

- 1) Scan the database one or more times to construct the initial tree. Restructure the tree using various methods to compact it.
- 2) Apply the various designed pattern-growth approaches to identify the candidate HUIs.
- 3) From the candidate itemsets, identified the HUIs.

While the tree structure is compact, it may not consistently be minimal and could occupy a considerable amount of memory space. The effectiveness of these mining algorithms is closely linked to two pivotal factors.

- ✓ **The Number of Conditional Trees:** The more conditional trees constructed during the mining process greatly impact on mining performance.
- ✓ **Construction/Traversal Cost:** The cost associated with constructing and traversing each conditional tree also plays a crucial role in determining mining performance.

These algorithms generate a numerous conditional trees for the huge database and low *MinUtility* threshold; there are challenges related to traversal cost and storage for conditional trees. So the conditional trees generation is the major performance bottlenecks of these algorithms, which results in high memory consumption and execution time. These tree-based approaches offer several advantages, as mentioned below:

- **Fewer Passes over the Dataset:** They require only two to three times database scan.
- **Data Compression:** These techniques effectively compress datasets into a tree structure.
- **Improved Speed:** These techniques are required considerably lower execution time than apriori-like approaches.

However, these tree-based approaches for HUIM have certain limitations:

- **Memory Constraints:** The generated tree may be too large to fit into memory.
- **Costly Tree Construction:** Building the tree can be resource-intensive.
- **Time-Consuming:** These tree-based approaches required more time to process all conditional prefix trees recursively.
- **Sensitivity to Parameters:** The *MinUtility* parameter is also affect the algorithm's performance.

3.4 List based approaches

In pursuit of greater efficiency compared to both apriori-based and tree-based methods, researchers have introduced an innovative data structure the "utility list" to manage item utility information. These approaches, which leverage utility lists, work in a single phase. They address the limitations of apriori-based and tree-based techniques.

3.4.1 HUI-Miner[54]

In 2012, Liu and Qu introduced the first single-phase high utility itemset mining approach, namely HUI-Miner. It eliminates the need of candidate generation. This breakthrough not only reduced mining time but also addressed the significant challenge of candidate generation that was prevalent in earlier apriori-based and tree-based approaches. Their innovation includes introducing a novel data structure called the "utility list," designed to store essential utility information of itemsets. Furthermore, to avoid the time-consuming process of mining HUIs, the algorithm employed an efficient pruning technique utilize the total of item's utility and the remaining utility. Novel utility list structure is a triplet $\langle TRid, iutility, rutility \rangle$ where $TRid$ represents the transaction ID of the transaction containing the itemset, $iutility$ is the itemset's utility, $rutility$ is the heuristic information that stores the utility value of the remaining itemset in the transaction.

□ Initial Utility List

The process of HUI-Miner begins with creating the Initial Utility List, which involves two-step database scanning to construct the initial utility list for 1-itemsets.

In the first database scan, the algorithm calculates the TWU for each item. During the second database scan, items that have a TWU below $MinUtility$ threshold are discarded. The remaining items are then sorted in ascending order of TWU in each transaction. These modified transactions are referred to as "revised transactions," and the database itself is termed a "revised database."

Table 3.3: Sample transaction database

Transaction	A	B	C	D	E	F	G
T1	-	1	2	-	1	-	-
T2	3	4	-	-	2	-	1
T3	1	2	3	4	5	-	-
T4	-	-	-	-	-	3	1
T5	1	1	-	1	-	-	-

Table 3.4: Sample utility table

Item	A	B	C	D	E	F	G
Profit	5	1	3	4	2	1	2

Table 3.5: Sample revised database

Transaction	C	D	E	A	B
T1	2	-	1	-	1
T2	-	-	2	3	4
T3	3	4	5	1	2
T4	-	-	-	-	-
T5	-	1	-	1	1

Consider the sample database mentioned in Tables 3.3 & 3.4, and the user-specified *MinUtility* threshold is 40. HUI-Miner scans the database and calculates the TWU of each item. The items F and G are unpromising as their TWU is lower than the *MinUtility* threshold. After discarding unpromising items, HUI-Miner scan the database again, and the items are arranged in TWU ascending order in each transaction. The TWU ascending order of the items mentioned in the example is C-D-E-A-B, as denoted in Table 3.5.

Definition 3.1: *The itemset x exist in the transaction T , the set of all items which is successor of itemset X in the Transaction T is denoted as $T|X$.*

For example, $T_3|CE = \{AB\}$ and $T_2|E = \{AB\}$.

Definition 3.2: Remaining Utility

For the transaction T and itemset X , where X is exist in the T . The remaining utility of an itemset X is the total utility of all the item's utility in $T|X$, and it is denoted as $rutility(X, T)$.

$$rutility(X, T) = \sum_{i \in (T|X)} u(i, T) \quad (3.1)$$

Where $X \subseteq T$, the set of all the items, comes after itemset X in the transaction, and it is represented as $T | X$.

For example $rutility(CD, T_3) = 17$ and $rutility(E, T_2) = 19$.

During the second time database scanning, HUI-Miner constructs the initial utility lists of all promising 1-itemsets which are represented in Table 3.6. The basic element of the utility list is the triplet $\langle TRid, iutility, rutility \rangle$ where $TRid$ is a transaction identification number of the transaction which contains the itemset, $iutility$ is the itemset utility in the transaction, and $rutility$ is the remaining utility of the itemset in the transaction. For example, consider itemset E; three elements are added to the utility list of itemset E as three transactions T_1 , T_2 , and T_3 contain itemset E. Utility of E in T_1 is the $u(E, T_1) = 2$, and the remaining utility of E in T_1 is $u(B, T_1) = 1$ so the element $\langle T_1, 2, 1 \rangle$ is added in the utility list. Similarly, $\langle T_2, 4, 19 \rangle$ and $\langle T_3, 10, 7 \rangle$ are added to the utility list of 1-itemset E.

Table 3.6: Initial utility list of 1-itemset

{C}			{D}			{E}		
TRid	iutility	rutility	TRid	iutility	rutility	TRid	iutility	rutility
T1	6	3	T3	16	17	T1	2	1
T3	9	33	T5	4	6	T2	4	19
						T3	10	7

{A}			{B}		
TRid	iutility	rutility	TRid	iutility	rutility
T2	15	4	T1	1	0
T3	5	2	T2	4	0
T5	5	1	T3	2	0
			T5	1	0

□ Utility list of 2-itemset

The utility lists of two 1-itemsets are joined to create a utility list of 2-itemset without scanning the dataset. The utility list join operation identifies common transactions from both the utility lists and adds the elements $\langle TRid, iutility, rutility \rangle$ where $TRid$ is the common transaction ID, $iutility$ is the summation of utility of two items in the common transaction, and $rutility$ is the $rutility$ values of the successor item as per total order. i.e., the utility list of 2-item set $\{CD\}$ is constructed to find out the common transaction T_3 from the utility lists of $\{C\}$ and $\{D\}$. Add the element in the utility list of $\{CD\}$ as $TRid$ is T_3 , set the $iutility$ value is a 25, it is the sum of $iutility$ values from the utility list of $\{C\}$ and $\{D\}$, and the $rutility$ value is 17 as $rutility$ value of $\{D\}$ because D comes after $\{C\}$ as per TWU ascending order. Table 3.7 shows the utility lists of 2-itemsets

Table 3.7: Utility list of 2-itemset

{CD}			{CE}			{CA}		
TRid	iutility	rutility	TRid	iutility	rutility	TRid	iutility	rutility
T3	25	17	T1	8	1	T3	14	2
			T3	19	7			

{CB}		
TRid	iutility	rutility
T1	7	0
T3	11	0

□ Utility list of k-itemset

The utility list of k-itemset $(i_1 i_2 i_3 \dots i_k)$ denoted as itemset P_{xy} can be constructed by performing join operation on two utility lists of $(k-1)$ -itemset denoted as P_x and P_y . It conducts the join operation the same as generating the utility list of 2-itemset. It calculates the $iutility$ value of an element as $u(P_{xy}, T) = u(P_x, T) + u(P_y, T) - u(P, T)$. As the itemset P_x and P_y are the extension of itemset P , the utility of the P is accommodated in both the utility of P_x and P_y , so the utility of P is subtracted from the equation. The $rutility$ value is set as $rutility$ value of P_y . For example, in the construction of the $\{CDE\}$'s utility list, the element

$\langle T3, 35, 7 \rangle$ can be added to the itemset $\{CDE\}$'s utility list. Similarly, all the elements are added to the utility list of $\{CDE\}$.

❑ Search space and Pruning

The search space of the HUI-miner is visualized as a set enumeration tree. Within this tree structure, an itemset is referred to as an extension of the itemset represented by its parent node. Specifically, an itemset labeled as $(k+i)$ signifies an i -extension of the itemset labeled as k , and it is positioned in the tree as a descendant of its ancestor node. For example, in Figure 3.3, itemsets $\{CDE\}$ and $\{CDA\}$ are the 1-extensions of $\{CD\}$ while $\{CDEA\}$ is the 2-extension of $\{CD\}$.

HUI Miner algorithm starts from the root node to construct itemset and check all its 1-extensions. Then, it recursively creates the promising extension of itemsets, applying the joining of utility lists and pruning strategies. If the sum of *utility* and *rutility* values in the utility list of an itemset is less than *MinUtility* threshold, then it is pruned. The itemset cannot be further extended as it is an unpromising itemset.

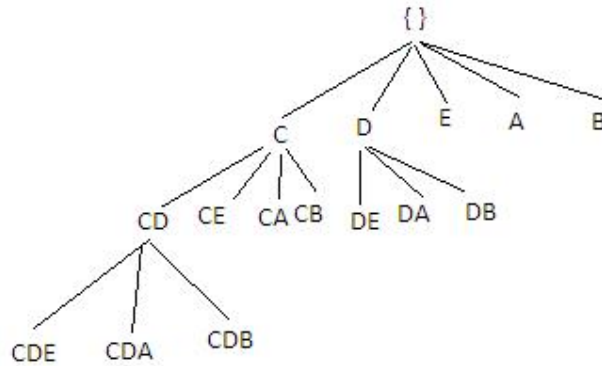


Figure 3.3: Set enumeration tree

Property 3.1: Consider the X' is the extension of itemset X then $(X' - X) = (X / X')$

Rationale: The itemset X is combined with the item(s) after X according to total order is the extension of X called X'

Lemma 3.1: Consider the itemset X , the total of *utility* and *rutility* values of all elements in the utility list of X is lesser than *MinUtility* threshold, then X' an extension of X is not a high utility itemset [54].

i.e. consider the utility list of itemset $\{CE\}$ sum of all *iutility* and *rutility* from its utility list is 35, which is less than *MinUtility* (40), so it is pruned because any extension of $\{CE\}$ does not become a high utility itemset.

In general, HUI-Miner discovers the HUI sets without generating the candidate itemsets. It performs the pruning based on sum of *iutility* and *rutility*, which is a little tighter compared to TWU. The main limitation is that it performs the unnecessary costly utility list join operations for the itemsets that do not exist in the dataset. It also performs the join operations for the low-utility itemsets.

3.4.2 mHUI-Miner[55]

In 2017, Peng, koh, and Riddle proposed a tree structure to avoid the generation of utility lists of the itemsets that do not exist in the database. The HUI Miner[54] and FHM[56] explore the search space using a set enumeration tree and construct the utility list of some itemsets which does not exist in the database due to that these algorithms are somewhat inefficient. The modified HUI-Miner incorporates a tree structure called IHUP to HUI-Miner. According to the property of the IHUP Tree Structure, the pathway of the tree shows the transactions in the database, so all the information about the items is stored in the tree together. This information helps to identify the itemsets or patterns that are present in the database. Therefore, by gradually extracting itemsets along the tree's pathways, we can prevent the expansion of current itemsets into ones that are not present in the database.

The algorithm scans the database to calculate the TWU of each individual item and to discard unpromising items according to TWDC property. It examines the database again to construct a global IHUP Tree, header table, and utility list of each promising item. During the tree construction, the items are read from the transaction in TWU descending order, and the path is added in the tree. The header table contains the promising items and TWU values. The items in the header table are arrange in TWU descending order. The main purpose of the IHUP tree is to guide the expansion process of itemsets. Consider the database presented in Tables 3.3 & 3.4. mHUIMiner scans the database two times and constructs global IHUP Tree and Utility lists, as demonstrated in Figure 3.4.

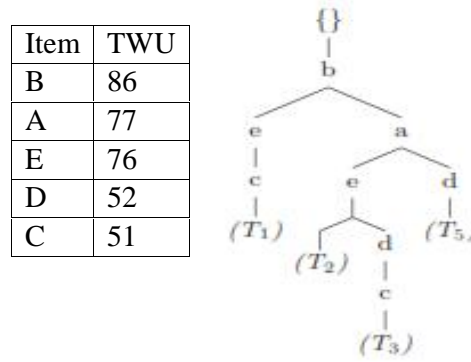


Figure 3.4: Global IHUP Tree

It recursively performs the mining procedure, selecting the item from the bottom of the header table. If the sum of the *iutility* and *rutility* is higher than the *MinUtility* value, then it constructs a local prefix tree. Considering item C, sum of *iutility* and *rutility* is 51, which is greater than *MinUtility* threshold i.e., 40 from its utility list, so creates a local prefix tree and header table of item C as presented in Figure 3.5, and calls the mining algorithm for that itemset p. Here the mining algorithm is a call for itemset $p = \{ C \}$.

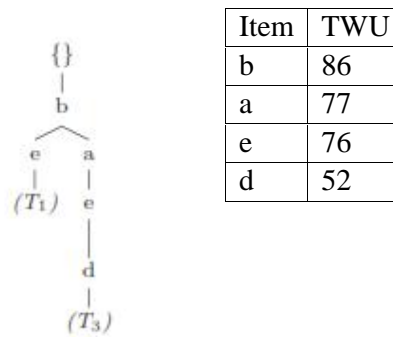


Figure 3.5: Local Prefix Tree for Tc

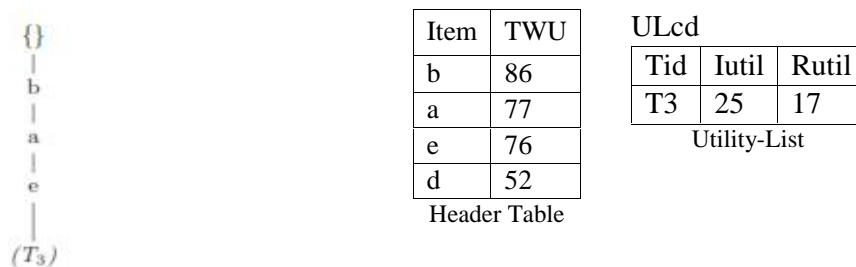


Figure 3.6: Local Prefix Tree for Tcd

The mining algorithm checks the sum of *utility* value from the utility list of item set p . If it is not less than *MinUtility* value, then the itemset p is added to the High Utility Itemset, and check the sum of *utility* and *rutility* values to further explore the itemset p . Here for itemset $p=\{C\}$ the sum of *utility* and *rutility* value is 41, so it is further explored with selecting the item $x=\{D\}$ from the bottom of its header table. Create a local prefix tree $T_{px} = T_{CD}$ is shown in Figure 3.6, and a utility list of itemset $px = \{CD\}$ by joining the utility lists of two itemsets according to HUI-Miner algorithm. The mHUI-Miner recursively calls the mining algorithm to discover all high utility itemsets.

Integrating the IHUP Tree structure to avoid the generation of itemsets that do not exist in the database can increase the performance of the High Utility itemset mining. It reduces the number of utility list join operations by guiding the itemset exploration process.

3.4.3 ULB-Miner[57]

In 2017, Duong, Viger, ramampiaro, norvag and dam proposed efficient high utility itemset mining using a buffered utility list called ULB-Miner. Author proposed a novel list structure called utility list buffer to store the item utility information and efficient join operation to generate a segment of itemsets in linear time. The ULB (Utility List Buffer) is based on the buffering utility information to reduce memory consumption. The ULB structure reuses the memory of the itemset that will not be further expanded.

□ The utility list buffer structure

Definition 3.3: Utility-list buffer structure

Let $i \in I$, set of items I in database D , which consists of set of transactions T . The utility list buffered structure is denoted as UL_{Buf} , a memory pipeline to store itemset information as utility lists. The UL_{Buf} holds a set of tuples of the form $\langle TRid, iutility, rutility \rangle$ where $TRid \in D$, $iutility \in R$, and $rutility \in R$, which is called data segments.

The index segments $SUL(x)$ are created to access the information of itemset x stored in the UL_{Buf} .

Definition 3.4: Summary of Utility List

Consider the database D , and itemset x , $SUL(x)$ is the index segment called a summary of utility list defined as a tuple $(x, StartPos, EndPos, SumIutility, SumRutility)$ where X is the

itemset, StartPos, and EndPos indicating the respective starting and ending positions of itemset x within the UL_{Buf} (Utility List Buffer) where utility information is stored. SumIutility signifies the cumulative sum of utility values for the itemset X , and SumRutility denotes the total of remaining utility values.

Definition 3.5: Summary List

It the memory pipeline of Sum of Utility Lists $SUL(X)$ and denoted as $SL(D) = \{SUL(x), x \in I\}$

The ULB-Miner algorithm scans the database first and calculates the TWU of all single items and, discards the unpromising items. During the second database scan, it inserts item utility information into the UL_{Buf} according to TWU ascending order. It simultaneously records the summary of the utility list into the summary list. The utility list buffer UL_{Buf} and Summary list SL of the database shown in Tables 3.3 and 3.4 is presented in Figure 3.7 and 3.8, respectively.

TRid	1	3	3	5	1	2	3	2	3	5	1	2	3	5
iutility	6	9	16	4	2	4	10	15	5	5	1	4	2	1
rutility	3	33	17	6	1	19	7	4	2	1	0	0	0	0

Figure 3.7: Utility list buffer for single items

SL=	Item=C	Item=D	Item=E	Item=A	Item=B
	StartPos=0	StartPos=2	StartPos=4	StartPos=7	StartPos=10
	EndPos=2	EndPos=4	EndPos=7	EndPos=10	EndPos=14
	SumIutility=15	SumIutility=20	SumIutility=16	SumIutility=25	SumIutility=8
	SumRutility=36	SumRutility=23	SumRutility=27	SumRutility=7	SumRutility=0

Figure 3.8: Summary of utility list for single items

The search procedure explores the search space from item $\{C\}$. It checks its *iutility* value; if it is higher than the *MinUtility* then it adds item $\{C\}$ into HUI sets. It extends the item $\{C\}$ by adding item $\{D\}$ if the sum of *iutility* and *rutility* of $\{C\}$ is no less than *MinUtility*. Then, it creates segments to store utility information of itemset $\{CD\}$ joining segments of $\{C\}$ and $\{D\}$. The procedure inserts these segments into the UL_{Buf} and maintains its summary in the summary list. Utility list buffer after inserting itemset $\{CD\}$ and SL is demonstrated in Figure 3.9 and Figure 3.10 respectively. The sum of *iutility* and *rutility* value 42 is greater than *MinUtility* (i.e. 40), so it considers a candidate to extend the search space. Next, the item $\{E\}$ is appended to itemset $\{C\}$ to generate itemset $\{CE\}$. The utility information of itemset $\{CE\}$ is inserted into the UL_{Buf} at positions 15 to 17, and an appropriate entry for $\{CE\}$ is created into the summary list. The sum of *iutility* and *rutility* value of $\{CE\}$ is $35 <$

MinUtility. Thus, the entry of {CE} in UL_{Buf} and SL is cleared to further utilize the memory for another itemset. In the same way, items A and B are appended to item C, and the utility information of CA and CB is inserted into UL_{Buf} and SL, respectively. The below Figures 3.9 and 3.10 represent the UL_{buf} and SL after proceeding with the itemset {CD}. Similarly, after proceeding with the itemset {CA}, the UL_{buf} and SL are shown in Figures 3.11 and 3.12.

TRid	1	3	3	5	1	2	3	2	3	5	1	2	3	5	3
iutility	6	9	16	4	2	4	10	15	5	5	1	4	2	1	25
rutility	3	33	17	6	1	19	7	4	2	1	0	0	0	0	17

Figure 3.9: Utility list buffer after inserting itemset {CD}

Item=C StartPos=0 EndPos=2 SumIutility=15 SumRutility=36	Item=D StartPos=2 EndPos=4 SumIutility=20 SumRutility=23	Item=E StartPos=4 EndPos=7 SumIutility=16 SumRutility=27	Item=A StartPos=7 EndPos=10 SumIutility=25 SumRutility=7	Item=B StartPos=10 EndPos=14 SumIutility=8 SumRutility=0	Item=CD StartPos=14 EndPos=15 SumIutility=25 SumRutility=17
--	--	--	--	--	---

Figure 3.10: Summary of Utility list after inserting itemset {CD}

TRid	1	3	3	5	1	2	3	2	3	5	1	2	3	5	3	3
iutility	6	9	16	4	2	4	10	15	5	5	1	4	2	1	25	14
rutility	3	33	17	6	1	19	7	4	2	1	0	0	0	0	17	2

Figure 3.11: Utility list buffer after inserting itemset {CA}

Item=C StartPos=0 EndPos=2 SumIutility=15 SumRutility=36	Item=D StartPos=2 EndPos=4 SumIutility=20 SumRutility=23	Item=E StartPos=4 EndPos=7 SumIutility=16 SumRutility=27	Item=A StartPos=7 EndPos=10 SumIutility=25 SumRutility=7	Item=B StartPos=10 EndPos=14 SumIutility=8 SumRutility=0	Item=CD StartPos=14 EndPos=15 SumIutility=25 SumRutility=17	Item=CA StartPos=15 EndPos=16 SumIutility=14 SumRutility=2
--	--	--	--	--	---	--

Figure 3.12: Summary of Utility list after inserting itemset {CA}

TRid	1	3	3	5	1	2	3	2	3	5	1	2	3	5	3	1	3
iutility	6	9	16	4	2	4	10	15	5	5	1	4	2	1	25	7	11
rutility	3	33	17	6	1	19	7	4	2	1	0	0	0	0	17	0	0

Figure 3.13: Utility list buffer after inserting itemset {CB}

Item=C StartPos=0 EndPos=2 SumIutility=15 SumRutility=36	Item=D StartPos=2 EndPos=4 SumIutility=20 SumRutility=23	Item=E StartPos=4 EndPos=7 SumIutility=16 SumRutility=27	Item=A StartPos=7 EndPos=10 SumIutility =25 SumRutility =7	Item=B StartPos=10 EndPos=14 SumIutility =8 SumRutility=0	Item=CD StartPos=14 EndPos=15 SumIutility=25 SumRutility =17	Item=CB StartPos=15 EndPos=17 SumIutility =18 SumRutility =0
--	--	--	--	---	---	--

Figure 3.14: Summary of Utility list after inserting itemset {CA}

The ULB miner recursively expands the itemset and discovers the HUIs. It also efficiently joins segments of utility lists in UL_{Buf} in linear time. For joining the utility list, it finds the common transaction from both utility lists.

In general, ULB-Miner efficiently utilizes the memory as it removes the itemset's utility list from the memory that is not further expanded according to the pruning policy. The other itemsets will utilize the memory space. The previous approaches store utility lists of all generated itemsets into the memory, even the itemsets are neither further extended nor high utility itemsets. ULB-Miner also performs the efficient join operation.

3.4.4 HUI-Miner*[58]

In 2019, Qu, Liu, and Viger proposed a HUI-Miner* to improve the performance of HUI-Miner by proposing a new structure called Utility-List*. It has been observed that in HUI-Miner, the utility list of k-itemsets is constructed using the *TRid* comparisons of both utility lists of (k-1)-itemsets, but all the *TRid* comparisons are not effective. The effective comparisons are matching of the *TRid*. The ineffective comparison leads to degrade the performance of the algorithm. The HUI-Miner* removes these ineffective comparisons by Utility-list* structure. The construction of Utility-List* is faster. The mining procedure of the HUI-Miner* is the same as the HUI-Miner. HUI-Miner* uses the Utility List* structure to store the utility information, and Utility List* is constructed horizontally. The elements belonging to similar transactions in the utility list* of different itemsets are linked together, so it helps to remove the ineffective comparisons. Hence, it improves the mining performance.

3.4.5 UBP-Miner[59]

In 2022, Peng Wu et al. proposed a novel bitwise operation, namely BEO (Bit mErge cOnstruction), for faster construction of utility list. It proposed a bit utility list for each itemset to maintain the information for an itemset that exists in the transaction. Unlike the conventional HUIM approaches, this algorithm performs faster utility list join operation by directly merging the bit utility lists of two itemset rather than searching the common transactions. Despite fast utility list construction, it requires extra memory to store the transaction bit and the mapping information of the *TRids* into the bit utility list. The bitset storing the transaction bit is maintained in the array due to variable-sized modern computer architecture that is time-consuming.

Table 3.8: Summary of utility list-based approaches for HUIM

Sr. No	Algorithm Name	Key points	Pros	Cons
1	HUI-Miner	Generates the itemsets by recursively exploring the search space. Creates a utility list of k-itemset by joining the utility lists of (k-1)-itemsets	One phase does not generate the candidate sets	Perform costly utility list Join operations. Constructs the utility lists of itemsets, which are not exist in the database.
2	mHUI-Miner	Explores the search space using the IHUP Tree.	Creates utility lists of only those itemsets that are available in the database.	Consume more memory to maintain sub-trees. It also takes more time to traverse the tree recursively.
3	ULB-Miner	Discards the unpromising itemset. Store the utility information into UL_{Buf} .	Memory efficient. Efficiently performs utility list join operation.	The utility list construction process is complicated. It performs costly utility list join operations.

4	HUI-Miner*	Horizontally constructs the Utility-List* of Item-Set. Elements of the itemsets associated with a transaction are linked together	Fast Utility list join operation due to effective comparison	Increases the running time for the large linked list
5	ULB-Miner	Uses the bit utility list. Constructs the utility list, merging the bit utility lists	The utility list construction process is faster.	Requires extra memory to store the transaction bit and mapping information of the TRids into the bit utility list

▪ Summary of utility list-based approaches

The utility list-based approaches for HUIM are recent and efficient in terms of execution time and memory utilization. Unlike the pattern growth and apriori-based approaches, these algorithms do not generate the candidate itemsets. The utility list stores the utility information and the search space is maintained in the set enumeration tree. They recursively expand the itemset by exploring the search space and construct the utility list of the expanded itemset by performing the join operation on the utility lists. Despite the execution time and memory efficiency, the performance of these algorithms is limited due to a number of costly utility list join operations. The cost of the join operations depends on the number of comparisons required to find the common transactions between the utility lists. The cost also depends on the number of join operations performed.

3.5 Research Gap

From the detailed literature survey, it has been identified that utility list-based approaches are performing better compared to other approaches. The performance of the utility list-based approaches is limited due to the costly utility list join operations. The performance of the HUIM algorithms can be improved by

- ✓ Utilizing the suitable data structure that can efficiently access and retrieve the utility information like utility list.
- ✓ Exploring the search space in such a way that can reduce the number of join counts and minimize the cost of the join operation by reducing the number of comparisons.
- ✓ Eliminating the unnecessary join operations.
- ✓ Utilizing the tight pruning measures that can minimize the candidate sets.
- ✓ Reducing the database scanning cost and frequency.

CHAPTER-4

SCAO based Search Space Exploration Technique for HUIM

4.1 Introduction

Based on a comprehensive review of the literature and an analysis of the problem's inherent characteristics, it has been determined that the efficiency of HUIM algorithms primarily depends on several factors. These factors include the dataset's attributes, strategies for exploring the search space, the data structure used to store utility information, the pruning criteria employed, and the mining techniques used. The literature reviews shows that over the past two decades, many approaches have been proposed for HUIM, utilizing various combinations of data structures, pruning methods, and mining strategies. Among these, the more recent utility list-based approaches have demonstrated superior performance in terms of execution time and memory utilization. However, the performance of the utility list-based methods is limited because these approaches perform costly utility list join operations. The cost of the join operations mainly depends on the number of join counts, and number of comparisons needed to search common transactions between the utility lists.

4.2 Search Space Exploration in HUIM

The search space in HUIM is the all possible itemsets that can be derived from each item in the transaction database. In simple term, the search space includes all the conceivable combinations of items known as itemsets, which can be generated from the dataset. HUIM (High Utility Itemset Mining) algorithms are designed to efficiently explore and traverse this extensive space to detect high utility itemsets based on predefined utility criteria. The set enumeration tree is used to represent the search space in HUIM [54]. The empty set $\{\}$ represents the root node of the tree. The tree exhibits a hierarchical structure, wherein each tier of nodes signifies itemsets that grow in size progressively. To illustrate, nodes at the first level represent a 1-itemset, nodes at the subsequent level signify pairs of items or a 2-itemset,

nodes at the following level indicate triplets, and so forth. For given itemset $I = \{i_1, i_2, i_3, \dots, i_n\}$, and the total order of items from I is $\{i_1 < i_2 < i_3 < \dots < i_n\}$, the set enumeration tree is built following the given steps: Initially, the empty root node is built. Next, n child nodes are formed from the root, each representing individual 1-itemsets in that order. Subsequently, the $(n-p)$ child nodes of the node representing the itemset $\{i_k, \dots, i_p\}$ where $1 \leq k \leq p < n$ are created. The child node represents the itemsets $\{i_k, \dots, i_p, i_{(p+1)}\}$, $\{i_k, \dots, i_p, i_{(p+2)}\}$, $\dots, \{i_k, \dots, i_p, i_n\}$. In the same way, recursively each level is constructed till entire leaf nodes are created. For example, consider the itemset $I = \{a, b, c, d\}$, and the total order of the items is $a < b < c < d$. The set enumeration tree for the example is shown in Figure 4.1

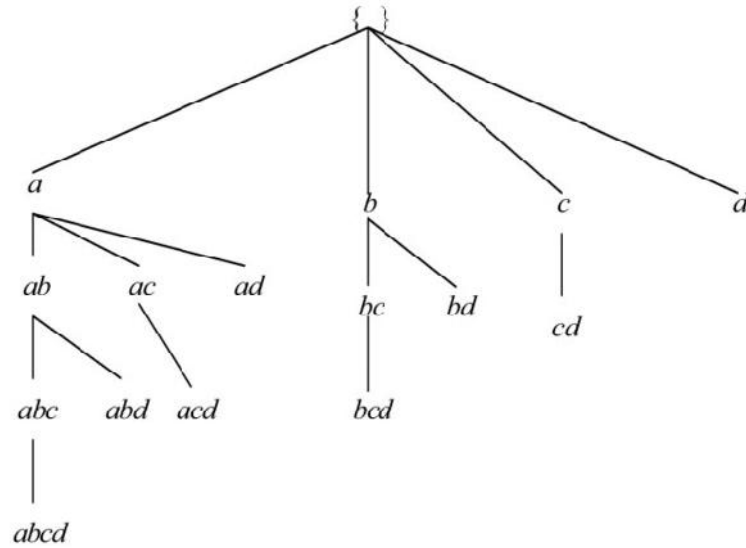


Figure 4.1: Set Enumeration Tree

The search space exploration is the traverse of the set enumeration tree to construct all possible itemsets. Search space exploration includes the order in which the nodes are built and the sequence to traverse the tree. The order is the same as the total order. The total order may be lexicography order, TWU ascending order, TWU descending order, etc. State-of-the-art HUIM approaches like HUI-Miner, mHUI-Miner, ULB-Miner, etc., explore the search space according to TWU ascending order to construct the itemset. Utility list-based approaches construct the utility list in the sequence same as the search space exploration sequence. One factor affecting the algorithm's efficiency is the number of comparisons required to find the common transactions for the utility list join operation. The search space exploration method is directed to number of comparisons required to find the common transactions between the utility lists needed for the join operation. This research work proposes a support count ascending order search space exploration sequence called SCAO-

based search space exploration. SCAO-based search space exploration method dramatically reduces the number of comparisons required to join the utility lists.

4.3 SCAO-based search space exploration

The proposed SCAO-based HUIM process model is presented in Figure 4.2. The proposed approach works in three phases.

Phase 1: Construction of a revised database

Scan the dataset to calculate the TWU and support count for each item. The support count of the itemset is the number of transactions containing the itemset. Unpromising items with a TWU lower than the *MinUtility* threshold are discarded. The database is scanned again to rearrange all transactions in ascending order based on their support count. The resulting set of revised transactions is referred to as the revised database. For instance, consider the data shown in Tables 4.1 and 4.2, with a *MinUtility* value 40. Items p and q are discarded as TWU of items fall below the *MinUtility* threshold. The items in each transaction are then rearranged in ascending order based on their support count, resulting in revised transactions such as m-n-o-k-l.

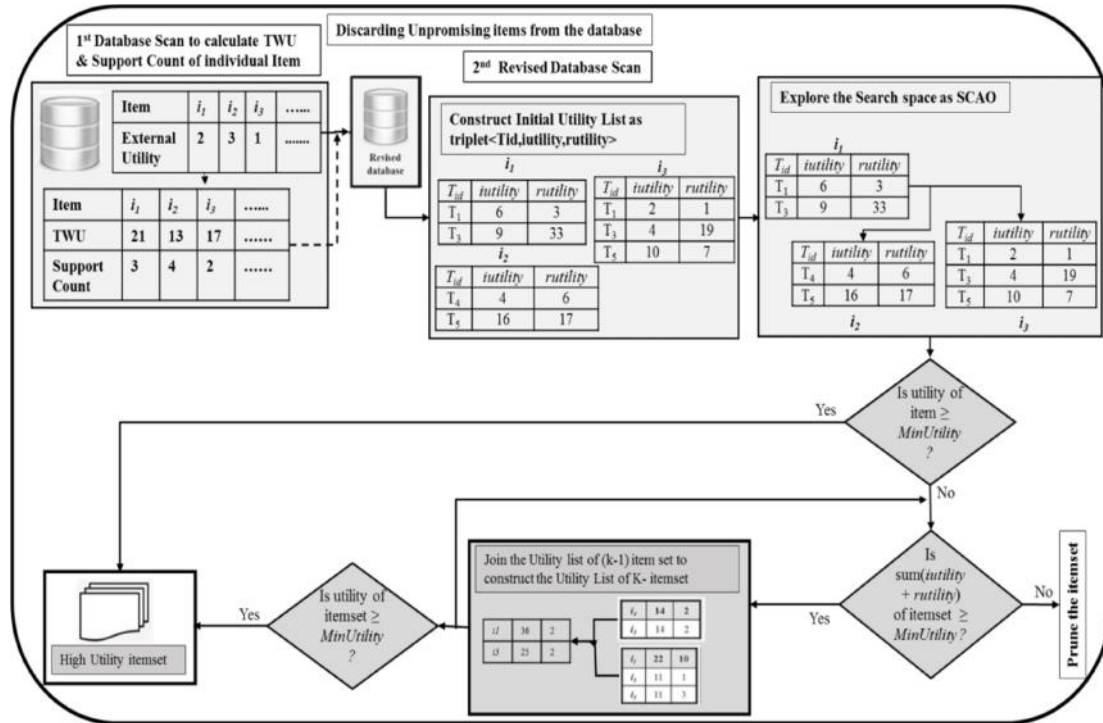


Figure 4.2: Proposed Model of SCAO-based search space exploration technique.

Table 4.1: Sample Transaction Database

Transaction	k	l	m	n	o	p	q
T ₁	-	1	2	-	1	-	-
T ₂	-	-	-	-	-	3	1
T ₃	3	4	-	-	2	-	1
T ₄	1	1	-	1	-	-	-
T ₅	1	2	3	4	5	-	-

Table 4.2: Sample Utility Table

Item	k	l	m	n	o	p	q
Profit	5	1	3	4	2	1	2

Phase 2: Construction of initial utility list

The database is scanned again to create the initial utility list for each promising item. The utility list of items consists of triplets associated with the transaction containing the item. Each triplet includes the following information: *TRid* (Item's transaction ID), *iutility* (Item's utility value in a revised transaction), and *rutility* (Item's remaining utility value) [54][55]

Table 4.3: Utility Lists of 1-Itemset

{m}			{n}			{o}		
TRid	iutility	rutility	TRid	iutility	rutility	TRid	iutility	rutility
T ₁	6	3	T ₄	4	6	T ₁	2	1
T ₅	9	33	T ₅	16	17	T ₃	4	19
						T ₅	10	7

{k}			{l}		
TRid	iutility	rutility	TRid	iutility	rutility
T ₃	15	4	T ₁	1	0
T ₄	5	1	T ₃	4	0
T ₅	5	2	T ₄	1	0
			T ₅	2	0

Definition 4.1: For the itemset X and transaction T , all the items in Transaction T that come after itemset X where $X \subseteq T$ is denoted as T/X . i.e $T_5 / mn = \{okl\}$ and $T_3/o = \{kl\}$

Definition 4.2: Remaining utility

The total utility of all the items that follow the itemset X in transaction T is represented as $rutility(X, T)$.

$$rutility(X, T) = \sum_{i \in (T \setminus X)} u(i, T) \text{ where } X \subseteq T \quad (4.1)$$

For example, constructing the item o 's utility list in transaction $T3$, $iutility$ value of o in $T3$ is 4 and remaining utility $rutility$ value of o in $T3 = rutility(o, T3) = u(k, T3) + u(l, T3) = 15 + 4 = 19$. Similarly, the initial utility list of all promising items (m, n, o, k, l) is constructed as shown in Table 4.3.

Phase 3: Search space exploration and mining process

The search space can be represented by a set enumeration tree depicted in Figure 4.3. Once the initial utility list is created, the proposed technique explores the search space in ascending order of support count called SCAO (support count ascending order) based search space exploration technique. It recursively extends the node in the tree representing the $(k-1)$ -itemsets by combining with its successor node until all possible itemsets are explored. Pruning is performed based on the sum of $iutility$ and $rutility$ to determine whether the itemset can be further extended. If the total of $iutility$ and $rutility$ values of all elements from the itemset's utility list is higher or equal to the $MinUtility$ threshold, then it can be further extended; otherwise, the itemset can be discarded. The specific details of the pruning mechanism are described in section 4.3.2. The utility lists of the k -itemsets are constructed according to algorithm 2 by joining the utility list of the $(k-1)$ -itemset and the utility list of its successor item. The details of the utility list join operation can be found in section 4.3.1. Additionally, the technique simultaneously discovers the high utility itemsets according to steps defined in the algorithm 1. For the itemset, before further extending the itemset, it is added to the set of HUIs if its sum of $iutility$ from the utility list is higher or equal to the user-specified $MinUtility$ threshold.

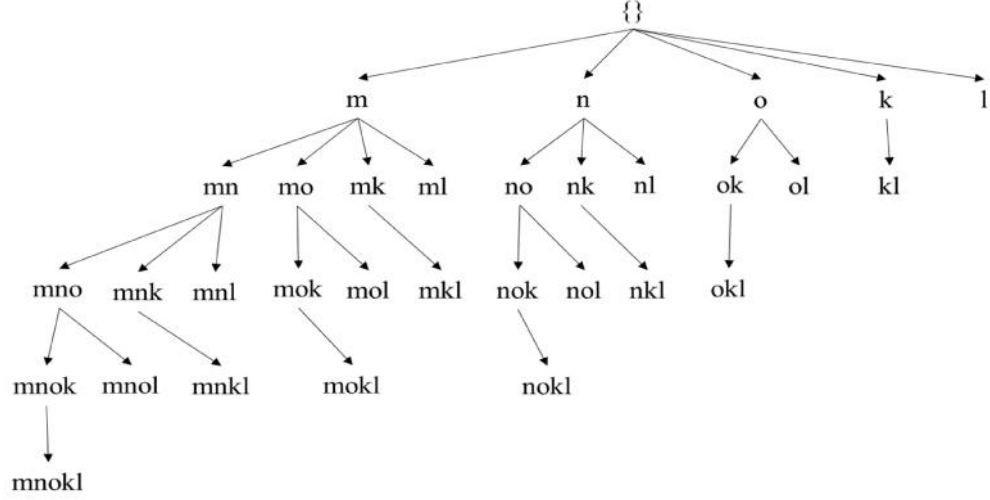


Figure 4.3: Set enumeration tree of a sample dataset.

4.3.1 A Construction of Utility list of 2-itemset and k-itemset

Consider the 2-itemset $x = \{mn\}$. To construct the utility list of itemset x , join operation is performed on the utility lists of m and n . During the join operation, a common transaction is searched from both utility lists and the element $\langle TRid, iutility, rutility \rangle$ is added to the utility list of $\{mn\}$. Here, $TRid$ represents the common transaction ID, $iutility$ represents the sum of $iutility$ values from utility lists of $\{m\}$ and $\{n\}$, and $rutility$ represents the $rutility$ value of $\{n\}$, as itemset $\{n\}$ comes after itemset $\{m\}$ according to SCAO. For example, when constructing the utility list of $\{mn\}$, the utility lists of m and n are joined. There is a common transaction with ID - T_5 , and the element $\langle T_5, 25, 17 \rangle$ is added to the utility list of $\{mn\}$. Similarly, add all the elements that have common transactions from the utility lists of $\{m\}$ and $\{n\}$ into the utility list of $\{mn\}$. Table 4.4 shows the utility lists of 2-itemsets. The utility list of a k -itemset can be constructed by joining the utility lists of two $(k-1)$ -itemsets. Performing join operation, the common transactions between two $(k-1)$ itemsets are identified, and the corresponding elements are added to the utility list of the k -itemset. Tables 4.5 and 4.6 represent the utility list of 3-itemset and 4-itemset, respectively. To identify the common transactions between utility lists, one needs to perform comparisons between the elements of the both utility lists. This process constructs utility lists for higher-order itemsets by leveraging the utility information from smaller itemsets.

Table 4.4: Utility Lists of 2-Itemset

{mn}			{mo}			{mk}		
TRid	iutility	rutility	TRid	iutility	rutility	TRid	iutility	rutility
T ₅	25	17	T ₁	8	1	T ₅	14	2
			T ₅	19	7			

{ml}			{no}			{nk}		
TRid	iutility	rutility	TRid	iutility	rutility	TRid	iutility	rutility
T ₁	7	0	T ₅	26	7	T ₄	9	1
T ₅	11	0				T ₅	21	2

Table 4.5: Utility Lists of 3-Itemset

{mno}			{mnk}			{mnl}		
TRid	iutility	rutility	TRid	iutility	rutility	TRid	iutility	rutility
T ₅	35	7	T ₅	30	2	T ₅	27	0

Table 4.6: Utility Lists of 4-Itemset

{mnok}			{mnol}			{mnokl}		
TRid	iutility	rutility	TRid	iutility	rutility	TRid	iutility	rutility
T ₅	40	2	T ₅	37	0	T ₅	42	0

Algorithm 1:- Mining Algorithm

Input: -LUL - List of UtilityList of 1-itemset

Prev.UL- previous item's utility list initially it is empty,

MU – User-specific Minimum utility threshold

Output: -high-Utility itemsets

1. For each element K in LUL do
2. If TOTAL of K's *iutility* \geq *MinUtility*, then
3. Add K & its previous itemsets in the resultset
4. endif
5. If the TOTAL of all K's *iutility* & *rutility* \geq MU, then
6. Create empty extUL;
7. For each element L follow K in LUL do
8. extUL=extUL+ join(Prev.UL, K, L)
9. endfor

```

10.      Mining(K, extUL, MU)
11.      endif
12.      Endfor

```

Algorithm 2:- Join Algorithm

Input: - UL_{Prev} - itemsetPrev 's utility list.

UL_K - itemset K's utility list,

UL_L - itemset L's utility list

Output: - UL_{KL} - itemset KL's utility list.

Initialize UL_{KL} is NULL

foreach component $E_k \in UL_K$

if \exists component $E_L \in UL_L \&\& E_k.TRid == E_L.TRid$ then

if UL_{Prev} empty then

Search component $E_{Prev} \in UL_{Prev}$ have the same T_{id}

Create new component

Where $E_{KL} = \langle E_k.TRid, E_k.iutility + E_L.iutility - E_{prev.iutility}, E_L.rutility \rangle$

else

$E_{KL} = \langle E_k.TRid, E_k.iutility + E_L.iutility - E_{prev.iutility}, E_L.rutility \rangle$

endif

Insert component E_{KL} to UL_{KL}

endif

endfor

return UL_{KL}

4.3.2 Pruning Mechanism

Search space exploration identifies all possible high utility itemsets, but it consumes more time because datasets have many items. Therefore, it is necessary to trim the search space by discarding the itemsets that do not contribute to the high utility. Use the itemset's *iutility* and *rutility* values from the itemset's utility list to narrow the search field. Only the itemset for which the total of its *iutility* and *rutility* values is no less than the *MinUtility* threshold is extended further. Otherwise, it can be discarded as per Lemma 1[16] because any of the superset of such itemset is not a high utility itemset. The total of *iutility* values from the itemset's utility list is the utility of the itemset. The itemset is a high utility itemset if its utility is no less than the *MinUtility* criterion.

Lemma 1:- If the sum of all iutility and rutility values from X's utility list $UL(X)$ is no less than the $MinUtility$, any itemset X' that is the extension of itemset X is not a high utility itemset.

i.e., consider the itemset $\{mn\}$'s utility list UL_{mn} , the total of itemset's *iutility* and *rutility* is 42, which is larger than the $MinUtility$ threshold 40. So, it can be further extended. While the total of *iutility* and *rutility* values of $\{mo\}$ is 35, it is discarded without further extension.

4.3.3 Analysis of Proposed Method Vs. State-of-the-art methods

The use of SCAO-based algorithms results in a reduced number of comparisons required to identify the common transactions between the utility lists for join operations. An analysis of time complexity reveals that the proposed SCAO-based algorithms require fewer comparisons when compared to other state-of-the-art methods for joining the utility lists. Let's consider the utility lists UL_a , UL_b , and UL_c , representing the utility values of 1-itemsets a , b , and c , respectively. The support counts for these utility lists are denoted as p , q , and r , respectively, with the condition that $p \geq q \geq r$. Now, for the construction of the utility list of itemset ab , perform a join operation between UL_a and UL_b .

Case 1: Consider the order sequence $a-b-c$ that is TWU ascending order of itemset a , b , and c . Existing state-of-art algorithms explore the search space as TWU ascending order therefore, they construct the utility lists in sequence as UL_{ab} , UL_{ac} , and UL_{abc} . Performing the join operations on UL_a and UL_b to construct UL_{ab} requires $q \log_2 p$ comparisons. Similarly for constructing UL_{ac} requires $r \log_2 p$ comparisons. The maximum number of possible entries in UL_{ab} is q , and in UL_{ac} is r .

UL_{abc} , Utilitylist of abc constructed by performing join operation on UL_{ab} and UL_{ac} . The number of comparisons is $r \log_2 q$. Therefore, total number of comparisons required is $q \log_2 p + r \log_2 p + r \log_2 q$.

The SCAO-based approach constructs the utility list in the sequence as UL_{cb} , UL_{ca} , and then UL_{cba} because the SCAO sequence is $c-b-a$. To construct UL_{cb} requires $r \log_2 q$, and to construct UL_{ca} requires $r \log_2 p$ comparisons. The maximum number of entries in UL_{cb} is r , and in UL_{ca} is also r . To construct the utility list UL_{cba} of itemset cba , number of comparisons required is $r \log_2 r$ to join UL_{cb} and UL_{ca} . Therefore, the total number of comparisons is $r \log_2 q$

+ $r \log_2 p + r \log_2 r$, which is lesser or equal to $q \log_2 p + r \log_2 p + r \log_2 q$ ($\because r \leq q$ and $r \leq p$ therefore $r \log_2 r \leq q \log_2 p$).

Case 2: TWU ascending order for itemset a, b and c is a-c-b. Existing state-of-art algorithms construct the utility lists in sequence as UL_{ac} , UL_{ab} , and UL_{acb} . So, the total number of comparisons is $r \log_2 p + q \log_2 p + r \log_2 q$. While the SCAO-based approach constructs the utility list in the sequence as UL_{cb} , UL_{ca} , and then UL_{cba} , so the total number of comparisons is $r \log_2 q + r \log_2 p + r \log_2 r$, which is lesser or equal to $r \log_2 p + q \log_2 p + r \log_2 q$ ($\because r \leq q$ and $r \leq p$ therefore $r \log_2 r \leq q \log_2 p$).

Case 3: TWU ascending order for itemset a, b and c is b-a-c. Existing state-of-art algorithms construct the utility lists in sequence as UL_{ba} , UL_{bc} , and UL_{bac} . So the total number of comparisons is $q \log_2 p + r \log_2 q + r \log_2 q$. While the proposed SCAO-based approach requires total $r \log_2 q + r \log_2 p + r \log_2 r$ comparisons, which is lesser or equal to $q \log_2 p + r \log_2 q + r \log_2 q$ ($\because r \leq q$ therefore $r \log_2 p \leq q \log_2 p$ and $r \log_2 r \leq r \log_2 q$).

Case 4: Consider the order sequence b-c-a presenting TWU ascending order of itemset a, b, and c. Existing state-of-art algorithms construct the utility lists in sequence as UL_{bc} , UL_{ba} , UL_{bca} . So, the total number of comparisons is $r \log_2 q + q \log_2 p + r \log_2 q$. While the Proposed SCAO based approach requires total $r \log_2 q + r \log_2 p + r \log_2 r$ comparisons, which is lesser or equal to $r \log_2 q + q \log_2 p + r \log_2 q$ ($\because r \leq q$ therefore $r \log_2 p \leq q \log_2 p$ and $r \log_2 r \leq r \log_2 q$).

Case 5: Consider the order sequence c-a-b that is TWU ascending order of itemset a, b, and c. Existing state-of-art algorithms construct the utility lists in sequence as UL_{ca} , UL_{cb} , and UL_{cab} . So, the total numbers of comparisons are $r \log_2 p + r \log_2 q + r \log_2 r$, which are the same as the proposed SCAO-based method.

Case 6: TWU ascending order for itemset a, b and c is c-b-a. The proposed SCAO-based algorithm performs Utility list join sequence in the same order c-b-a. Therefore, the numbers of comparisons are the same.

From all the above cases, it has been proved that the proposed join sequence as support count ascending order (SCAO) requires fewer comparisons compared to TWU ascending order. Hence, it reduces the cost of utility list join operation.

4.4 An illustrative example.

Consider the dataset tabulated in the Table 4.1 and Table 4.2. The user-specified *MinUtility* threshold is 40. The proposed SCAO-Based HUIM approach first scans the database and calculates the item's support count and TWU of all items, as shown in Table 4.7.

Table 4.7: Support count and TWU of item

Item	Support Count	TWU
k	3	77
l	4	86
m	2	51
n	2	52
o	3	76
p	1	5
q	2	30

Next, it discards the unpromising items; the item's TWU is less than the *MinUtility* threshold. So items p and q are discarded from the dataset and all the transactions are rearranged in support count ascending order, called the revised transaction. The set of revised transactions is called the revised dataset. The revised dataset for this example is shown in Table 4.8. The SCAO of all promising items becomes m-n-o-k-l.

Table 4.8: Revised Transaction dataset

	m	n	o	k	l
T ₁	2	0	1	0	1
T ₂	0	0	0	0	0
T ₃	0	0	2	3	4
T ₄	0	1	0	1	1
T ₅	3	4	5	1	2

By scanning the revised dataset, the initial utility list of each 1-itemset is constructed. The utility list of each item is shown in Table 4.3. Consider the item as a high utility itemset if its total utility from its utility list is not less than the *MinUtility* threshold. Next, it explores the search space maintained in the set enumeration tree, as shown in Figure 4.3. The itemsets is generated by extending them with its successor node from the tree in-depth search. So, it proceeds with the itemsets {m, mn, mno, mnok, mnokl.....} in sequence.

First, it constructs the utility list of itemset $\{mn\}$ by performing the join operation between m and n . The utility lists of the 2-itemset are shown in Table 4.4. For each generated itemset, it checks whether it is a HUIs; if so, adds the itemset to a set of HUIs. Here, the total of *utility* of $\{mn\}$ is 25, so there is no need to add in output. After that, it is decided whether the itemset is further extended based on the pruning measure sum of *utility* and *rutility*. Here the pruning measure sum of *utility* and *rutility* value of itemset $\{mn\}$ is 42 higher than the *MinUtility* threshold, so itemset $\{mn\}$ can be further extended by adding the item $\{o\}$. The utility lists of 3-itemset and 4-itemset are shown in Table 4.5 and Table 4.6, respectively. In the same way all itemsets are preceded and simultaneously added the itemset into the set of HUIs recursively. So the itemset construction order of the proposed approach is $\{m, mn, mno, mnok, mnokl, \dots\}$. Finally, it discovers the HUIs from the given dataset which is $\{mnok, mnokl, mnol, okl\}$.

4.5 Performance Evaluation

Existing state-of-the-art approaches for HUIM explore the search space as TWU ascending order. The proposed SCAO-based search space exploration technique incorporates into the existing state-of-the-art algorithms such as HUI-Miner, mHUI-Miner, and ULB-Miner. The proposed approaches are called SCAO-HUI-Miner, SCAO-mHUI-Miner, and SCAO-ULB-Miner. To evaluate the effectiveness of these approaches, extensive testing is conducted on diverse real datasets using different *MinUtility* percentages. The experimental results are then compared with other state-of-the-art methods.

4.5.1 Experimental Environment

All the algorithms were implemented in Java, and then tested on a system with 8GB RAM and an Intel Core i5 processor running Windows 10 Pro. To assess the performance of the algorithm under different *MinUtility* values, standard real-time datasets[60] were used. Table 4.9 provides details of the dataset properties and a comprehensive description. The datasets exhibited variations in the number of items, transactions, and transaction lengths.

Table 4.9: Characteristics of Dataset

Sr.No	Dataset Name	Number of Transactions	Number of Items	Average transaction length
1	Foodmart	4141	1559	4.4
2	Connect	67,557	129	43
3	Chess	3196	75	37
4	Retail	88,162	16,470	10.3
5	BMS	59,602	497	2.51
6	Kosark	990002	41270	8.1000
7	eCommerce	14975	3468	11.71

4.5.2 Performance Evaluation with HUI-Miner

The compared algorithms were executed on diverse datasets using progressively decreasing utility thresholds until either the execution time became excessive or the memory reached its capacity. The number of comparisons required for join operations and execution time were recorded during these experiments. The number of comparisons and running time of the proposed SCAO-HUI-Miner is compared with HUI-miner on various real datasets, which are shown in Tables 4.10 & Table 4.11 respectively. Number of comparisons and running time of the proposed SCAO-HUI-Miner and HUI-miner on various real datasets are plotted in the Figures 4.4 & Figure 4.5, respectively. The result analysis shows that the proposed SCAO-HUI-Miner requires 12 to 27 percent less number of comparisons compared to HUI-miner and it is 8 to 22 percent faster than HUI-Miner on standard real datasets.

Table 4.10: No of comparisons (HUI-Miner Vs SCAO-HUI-Miner)

Dataset	HUI-Miner (in Thousands)	SCAO-HUI-Miner (in Thousand)	Improvement (%)
Retail	3377	2458	27.21
Bms	41369	32648	21.08
Foodmart	44637	39426	11.67
kosark	78336	62348	20.41
Chess	734152	624573	14.93

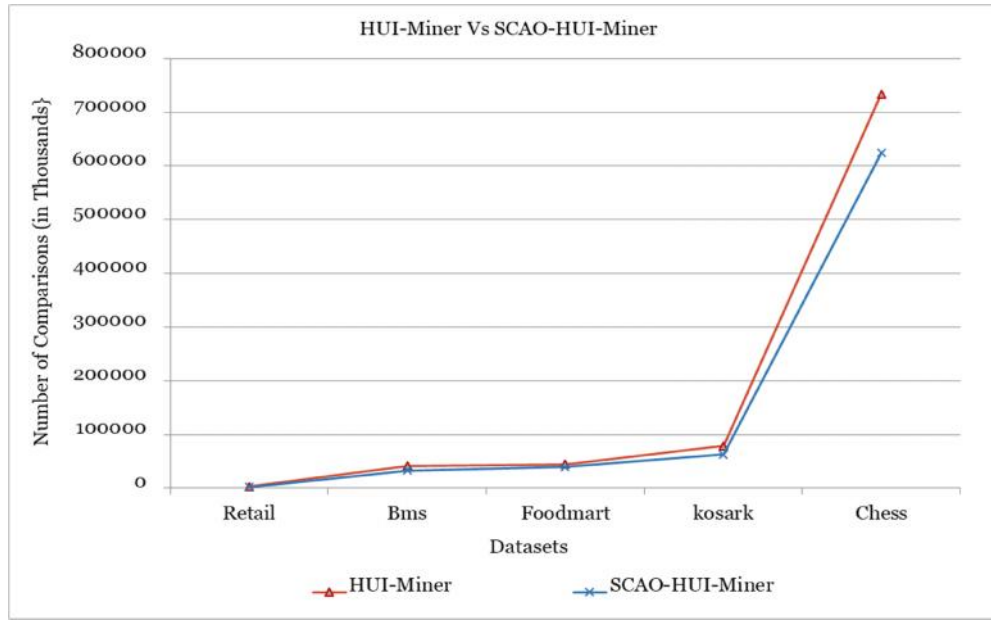


Figure 4.4: Number of comparisons (HUI-Miner Vs. SCAO-HUI-Miner)

Table 4.11: Execution time comparison (HUI-Miner Vs SCAO-HUI-Miner)

Dataset	HUI-Miner	SCAO-HUI-Miner	Improvement (%)
retail	617	479	22.37
bms	1088	904	16.91
Foodmart	1105	1019	7.78
chess	4139	3611	12.76
kosark	4420	3748	15.2

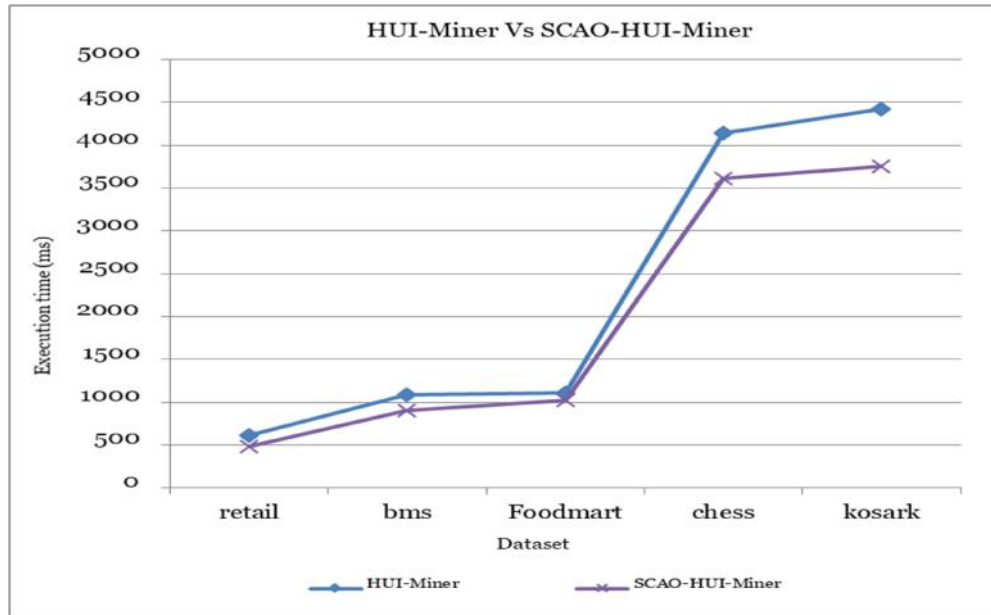


Figure 4.5: Execution time comparison (HUI-Miner Vs. SCAO-HUI-Miner)

4.5.3 Performance Evaluation with mHUI-Miner

The comparative analysis of number of comparisons and running time required for the proposed SCAO-based search space exploration employed to mHUI-Miner called SCAO-mHUI-Miner and mHUI-Miner is presented in Table 4.12 & Table 4.13. Performances of both approaches for number of comparisons and running time are plotted in Figure 4.6 & Figure 4.7. The proposed technique SCAO-mHUI-Miner requires approximately 10 to 25 percent less number of comparisons and 6% to 23% less execution time than mHUI-Miner.

Table 4.12: No of comparisons mHUI-Miner Vs SCAO-mHUI-Miner

Dataset	mHUI-Miner (in Thousand)	SCAO-mHUI-Miner (in Thousand)	Improvement (%)
Foodmart	1796	1379	23.22
retail	3255	2936	9.8
bms	37491	27939	25.48
kosark	75035	67087	10.59
chess	403105	324066	19.61

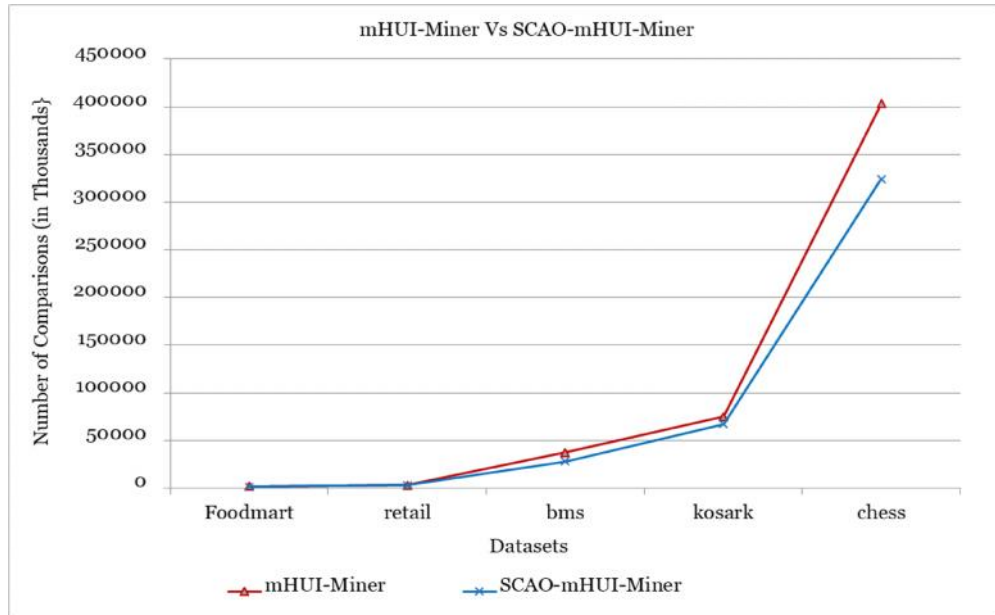


Figure 4.6: Number of comparisons (mHUI-Miner Vs SCAO-mHUI-Miner)

Table 4.13: Execution time comparison (mHUI-Miner Vs SCAO-mHUI-Miner)

Dataset	mHUI-Miner (Execution time in ms)	SCAO-mHUI-Miner (Execution time in ms)	Improvement (%)
Foodmart	225	177	21.33
bms	385	295	23.38
retail	566	528	6.71
chess	3067	2535	17.35
kosark	5241	4770	8.99

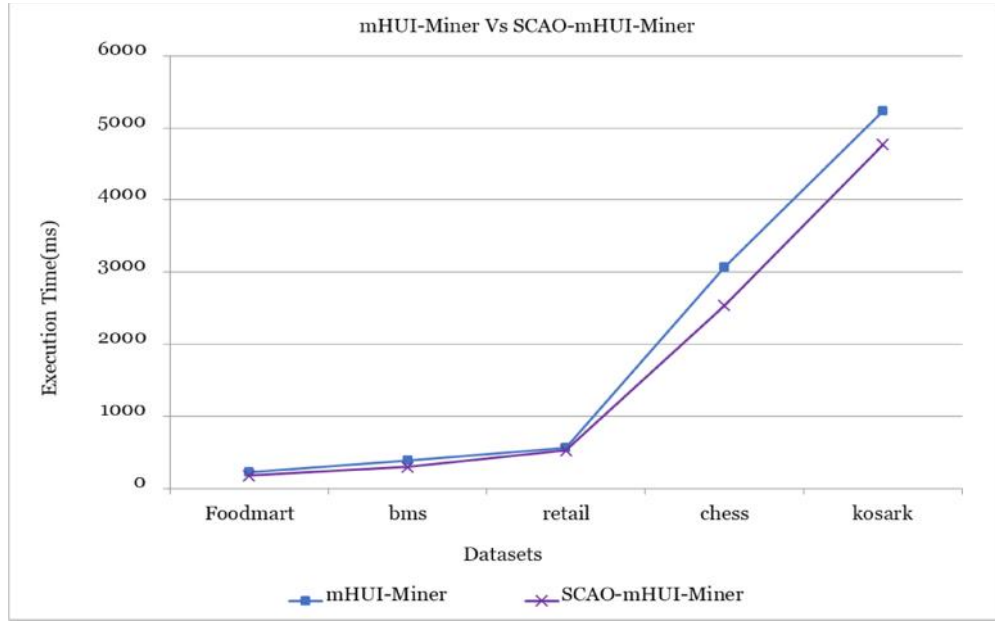


Figure 4.7: Execution time comparison (mHUI-Miner Vs SCAO-mHUI-Miner)

4.5.4 Performance Evaluation with ULB-Miner

The comparison of the proposed approach incorporated into ULB-Miner called SCAO-ULB-Miner and ULB-Miner for number of comparisons and running time is as shown in Table 4.14 & Table 4.15 and demonstrated in Figure 4.8 & Figure 4.9. The proposed technique SCAO-ULB-Miner has been found to be 12 to 28% faster than ULB-Miner because SCAO-ULB-Miner requires 8 to 26 percent less number of comparisons.

Table 4.14: No of comparisons ULB-Miner Vs SCAO-ULB-Miner

Dataset	ULB-Miner (in Thousand)	SCAO-ULB-Miner (in Thousand)	Improvement (%)
Foodmart	536	396	26.12
retail	2856	2249	21.25
bms	33284	30547	8.22
chess	37733	32695	13.35
kosark	67378	58423	13.29

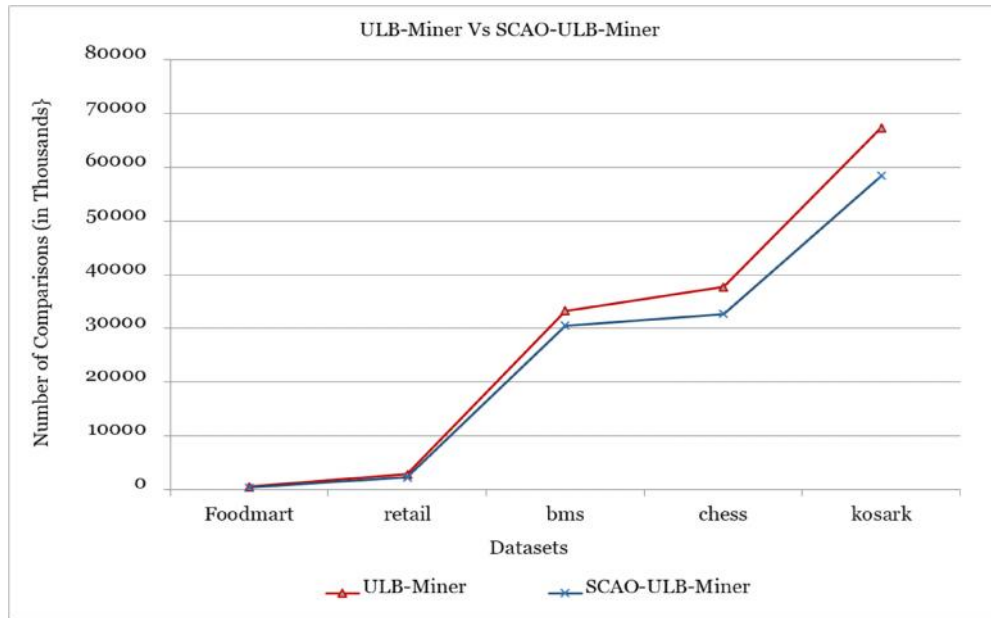


Figure 4.8: Number of comparisons (ULB-Miner Vs SCAO-ULB-Miner)

Table 4.15: Execution time comparison (ULB-Miner Vs SCAO-ULB-Miner)

Dataset	ULB-Miner (Execution time in ms)	SCAO-ULB-Miner (Execution time in ms)	Improvement (%)
Foodmart	322	231	28.26
retail	690	573	16.96
bms	829	680	17.97
chess	1170	1030	11.97
kosark	4797	4262	11.15

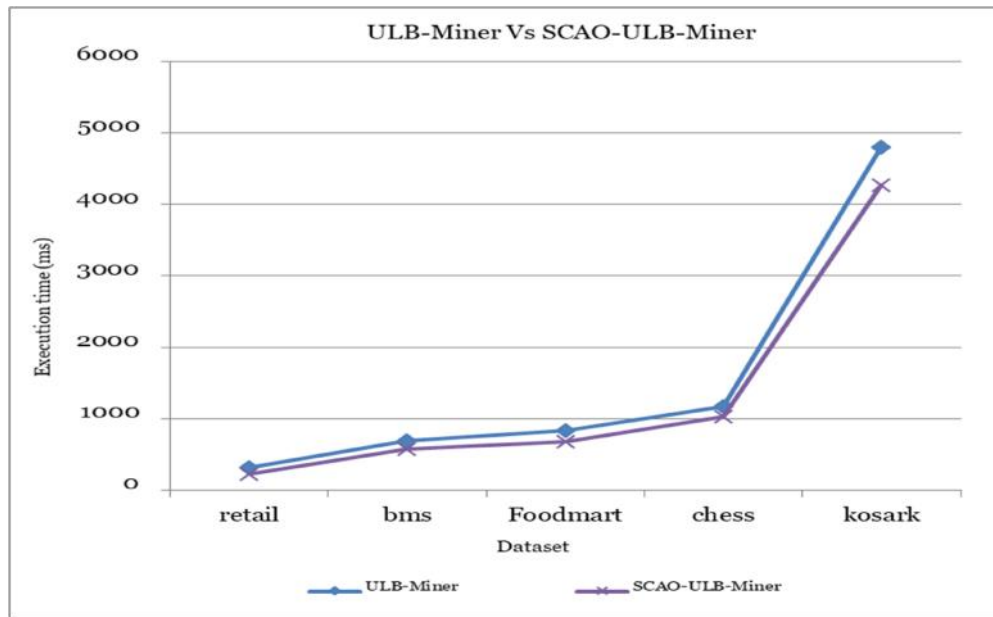


Figure 4.9: Execution time comparison (ULB-Miner Vs SCAO-ULB-Miner)

CHAPTER-5

PUCP-Miner: Predicted Utility Co-exist Pruning

5.1 Introduction

Utility list-based HUIM approaches perform several costly utility list join operations. From the literature survey, it has been identified that the performance of the existing state-of-the-art approaches is limited due to performing unnecessary utility list join operations. Based on the item's co-existence in the dataset, the researchers proposed Predicted Utility Co-Exist Structure known as PUCS to store the utility data. The research work also proposes Predicted Utility Co-Exist Pruning known as PUCP to eliminate unnecessary utility list join operations. PUCP greatly reduces the utility list join operations and thus it improves the performance of the algorithm. It eliminates the low utility itemsets directly without performing the join operations. Details of the proposed structure PUCS and proposed pruning method PUCP are described in the next section.

5.2 The PUCS (Predicted Utility co-exist Structure)

A novel structure called PUCS based on the coexistence analysis of the items is shown in Figure 5.1. The set of triplets of the form $(x, y, PU) \in I \times I \times R$ is known as PUCS. The predicted utility of itemset xy is represented by PU in the triplet. The PUCS is created concurrently with the creation of the initial utility list for the items during the second scanning of the database.

$$PU = \sum_{xy \in Ti} (iutility(xy) + rutility(xy)) \quad (5.1)$$

Figure 5.2 illustrates the PUCS structure of the sample dataset presented in Table 5.1 & Table 5.2. In this context, we define the pruning condition as “If there is no tuple (x, y, PU) in PUCS structure where PU is greater than or equal to $MinUtility$, then we consider the itemset $p = \{xy\}$ and its supersets are low utility itemsets. Consequently, there is no need to

further explore the itemset x , which means no need to perform the costly utility list join operation.

	L	M	N	O
K	PU_{KL}	PU_{KM}	PU_{KN}	PU_{KO}
L		PU_{LM}	PU_{LN}	PU_{LO}
M			PU_{MN}	PU_{MO}
N				PU_{NO}

Figure 5.1: PUCS Structure

	N	O	K	L
M	42	35	16	18
N		33	33	23
O			40	23
K				32

Figure 5.2: PUCS of the sample database

Table 5.1: Sample Transaction Database

Transaction	k	l	m	n	o	p	q
T_1	-	1	2	-	1	-	-
T_2	-	-	-	-	-	3	1
T_3	3	4	-	-	2	-	1
T_4	1	1	-	1	-	-	-
T_5	1	2	3	4	5	-	-

Table 5.2: Sample Utility Table

Item	k	l	m	n	o	p	q
Profit	5	1	3	4	2	1	2

During the second database scanning, the PUCS structure is constructed along with an initial utility list of items. Within the PUCS structure, we consider the elements for items M and N, represented as the triplet $\langle M, N, PU_{MN} \rangle$. Here, PU_{MN} corresponds to the sum of the *utility* and *rutility values* for the itemset $\{MN\}$. This value can be calculated during the initial database scanning process.

Additionally, we propose a unique pruning strategy called PUCP (Predicted Utility Co-exist Pruning) to minimize the number of join operations using the PUCS structure. This strategy aims to efficiently predict and remove itemsets that are unlikely to have significant utility, thereby reducing the computational burden associated with unnecessary join operations.

5.3 The PUCP (Predicted Utility co-exist Pruning)

According to property 1, previous algorithms like HUI-miner, mHUI-Miner, and ULB-Miner trim the search space, using the addition of *utility* and *rutility* values of an itemset. Existing algorithms explore the search space as TWU ascending order. The details of the search space exploration are included in chapter 4. Consider the item x , it can be further extended by its successor node to represent the item y . Existing approaches perform the join

operations on utility lists of x and y to construct the utility list of $\{xy\}$ even though itemset $\{xy\}$ is a low utility itemset. Existing state of art algorithms perform number of costly utility list join operations for constructing low-utility itemsets.

The proposed PUCP suggests whether the join operation should be performed or not. It eliminates the joining operations for the low utility itemset. For constructing the utility list of itemset $\{xy\}$, the proposed algorithm called PUCP-Miner checks the element $\langle x, y, PU_{xy} \rangle$ from PUCS. If an element does not exist in the PUCS where $PU_{xy} \geq MinUtility$, itemset $\{xy\}$ is discarded directly without constructing the utility list of itemset $\{xy\}$. As a result, it will minimize join operations of the utility list.

Consider *MinUtility* threshold value 40 as an example. To construct the itemset $\{MN\}$ and its utility list, join operation is applied on the utility list of individual items M & N . According to PUCP, it checks the PU_{MN} from the PUCS that is 42, so join operations have been performed. While for the construction of itemset $\{MO\}$ the PU_{MO} from the PUCS is 35, less than the *MinUtility* threshold, so there is no need to perform the join operation. By removing needless join operations, this pruning method allows PUCP to reduce the number of join operations significantly.

The overall procedure to discover high utility itemsets by the proposed approach namely PUCP-Miner is represented in figure 5.3.

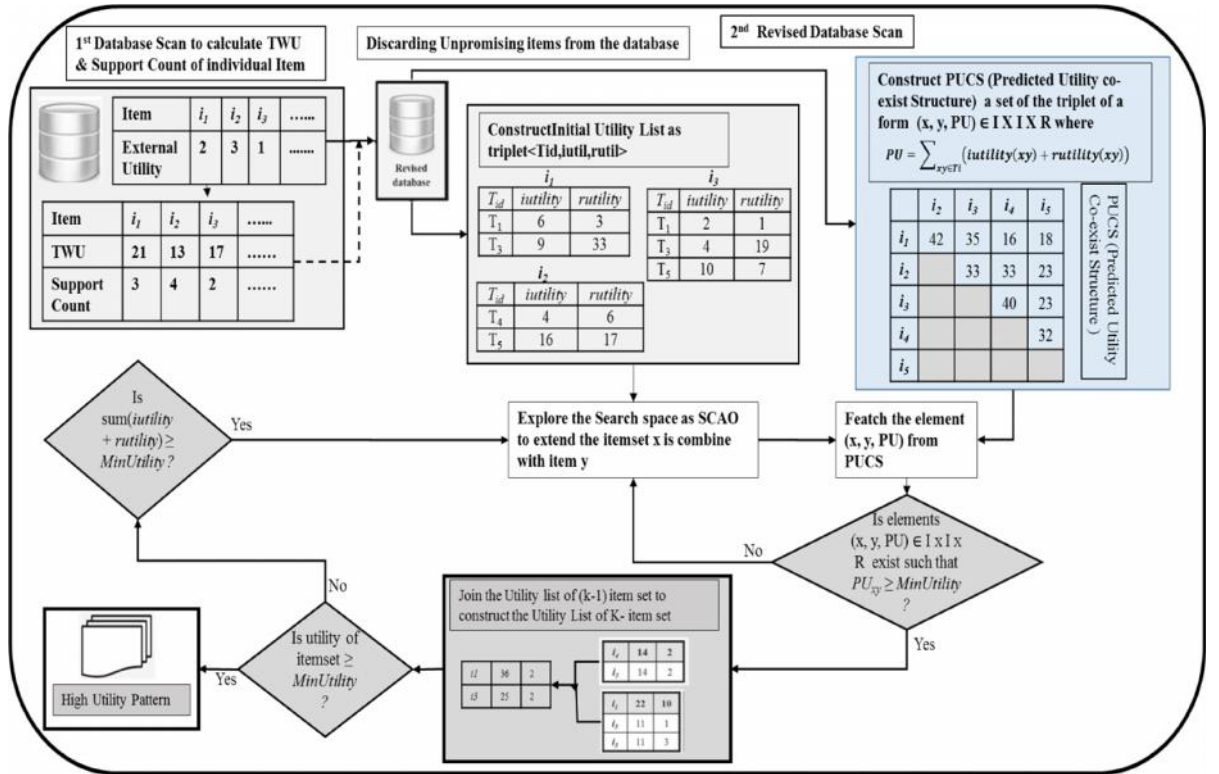


Figure 5.3: Proposed Model for PUCP-Miner.

First PUCP-Miner scans the dataset to calculate the TWU and support count of each item and discards the unpromising items. The items with TWU less than the *MinUtility* threshold are considered as unpromising items. After that, it arranges the remaining items in the transaction according to support count ascending order (SCAO) called a revised transaction. The set of revised transactions is called the revised database. The revised database is scanned and it generates the initial utility list of each item and constructs the PUCS simultaneously. The method then explores search space represented in the set enumeration tree as support count in ascending order to extend the itemset by combining with another item. Next, it fetches the element from the PUCS corresponding to the itemset to be extended. If the PU of the element satisfies the *MinUtility* requirement then the join operation is performed otherwise, the join operation is eliminated. The utility of the extended itemset is checked if it is higher or equal to *MinUtility* then it will be added to the HUI list. Next, it checks the pruning condition, if the *iutility* and *rutility* of the itemset are higher than the *MinUtility* threshold then the itemset is further extended. This procedure is performed recursively to explore all itemsets. Proposed methodology is summarized in Algorithm -1, 2, and 3.

Algorithm 1: Creating preliminary utility lists & Build PUCS

Input: - Transaction Database DB, Minimum utility threshold *MinUtility*

Output: - List of Utilitylist of each promising items LULs

1. Scan the DB
2. compute TWU of all items *i*
3. Calculate Support_Count of each item *i*
4. If $TWU(i) < MinUtility$ then
 Discard item *i*
Endif
5. Rearrange the items in the transaction as SCAO (Support Count Ascending Order)
6. Scan the database DB again
7. Create first list of utility for each favorable item.
8. Build the PUCS (Predicted Utility Co-exist Structure)

Algorithm 2: Mining Algorithm

Input: -LUL - List of UtilityList of 1-itemset

Prev.UL- previous item's utilitylist initially it is empty,

MinUtility – User-specific threshold for Minimum utility

Output: - itemsets with high utility

1. For each element K in LUL do
2. If TOTAL of K's *iutility* \geq *MinUtility* then
3. Add K & its previous itemsets in the resultset
4. End If
5. If TOTAL of all K's *iutility* & *rutility* \geq *MinUtility* then
6. Create empty extUL;
7. For each element L follow K in LUL do
8. If $\exists (K, L, PU_{KL}) \in PUCS$ such that
 $PU_{KL} \geq$ *MinUtility* then
extUL=extUL+ join(Prev.UL,K,L)
Endif
9. Endfor
10. Mining(K, extUL, *MinUtility*)
11. Endif
12. Endfor

Algorithm 3: Join Algorithm

Input: - UL_{Prev} - itemset Prev 's utility list.

UL_K – itemset K's utility list,

UL_L – itemset L's utility list

Output: - UL_{KL} - itemset KL's utility list.

1. Initialize UL_{KL} is NULL
2. Eoreach component $E_k \in UL_K$
3. if \exists component $E_L \in UL_L$ && $E_k.TRid == E_L.TRid$ then
4. if UL_{Prev} empty then
5. Search component $E_{Prev} \in UL_{Prev}$ have same TRid
6. Create new component
Where $E_{KL} = \langle E_k.TRid, E_k.iutility + E_L.iutility - E_{prev}.iutility, E_L.rutility \rangle$
7. else
8. $E_{KL} = \langle E_k.TRid, E_k.iutility + E_L.iutility - E_{prev}.iutility, E_L.rutility \rangle$
9. Endif
10. Insert component E_{KL} to UL_{KL}
11. Endif
12. Endfor
13. return UL_{KL}

5.4 Performance Evaluation

Comprehensive experiments were conducted on a variety of real datasets, employing different *MinUtility* percentages, to evaluate the performance of the proposed PUCP-Miner thoroughly. The experimental results of PUCP-Miner were compared to state-of-the-art methods including mHUI-Miner, ULB-Miner, and HUI-Miner, specifically focusing on memory requirements and execution time. Each experiment was implemented in a JAVA environment and executed on a computer with 8GB RAM and a 3.4GHz Intel Core i5 CPU. Standard real-time datasets were utilized in the experiments to measure performance of the algorithm accurately. Characteristics of the datasets[60] used in the experiments are presented in Table 5.3.

Table 5.3: Dataset characteristics used in experiments

Sr.No	Name of Dataset	Number of Transactions	Number of Items	Average transaction length
1	Foodmart	4141	1559	4.4
2	Retail	88,162	16,470	10.3
3	BMS	59,602	497	2.51
4	Kosark	990002	41270	8.1000
5	ecommerce	14975	3468	11.71

5.4.1 Execution Time Analysis.

The execution time of the proposed approach PUCP-Miner with state-of-the-art approaches HUI-Miner, mHUI-Miner, and ULB-Miner on different datasets is listed in Tables 5.4, 5.5, and 5.6, respectively. Execution time is also plotted in Figures 5.4, 5.5, and 5.6 for performance comparison. It has been observed that on the ecommerce dataset, compared to HUI-Miner, mHUI-Miner, and ULB-Miner, the suggested PUCP-Miner is almost 62%, 65%, and 20% faster, respectively. On the BMS dataset, PUCP-Miner takes nearly 45% less time than HUI-Miner, 46% less time than mHUI-Miner, and 42% less time than ULB-Miner. On the Foodmart dataset, PUCP-Miner is almost 67% quicker than HUI-Miner, 18% quicker than mHUI-Miner, and 18% quicker than ULB-Miner. On the retail dataset, the proposed PUCP-Miner almost takes 88%, 74%, and 35% less running time than HUI-Miner, mHUI-Miner, and ULB-Miner respectively. Lastly, on the Kosark dataset, PUCP-Miner is

approximately 32% faster than HUI-Miner, 25% faster than mHUI-Miner, and 27% faster than ULB-Miner. From the execution time analysis, it is concluded that the proposed PUCP-Miner outperforms on retail, BMS, and ecommerce datasets and it has a satisfactory improvement in execution time on Foodmart and Kosark datasets.

Table 5.4: Execution time comparison HUI-Miner Vs PUCP-Miner

Dataset	HUI-Miner (Execution time in ms)	PUCP-Miner (Execution time in ms)	Improvement (%)
Foodmart	393	131	66.67
BMS	394	218	44.67
ecommerce	1323	497	62.43
kosark	4415	3020	31.6
Retail	37857	4471	88.19

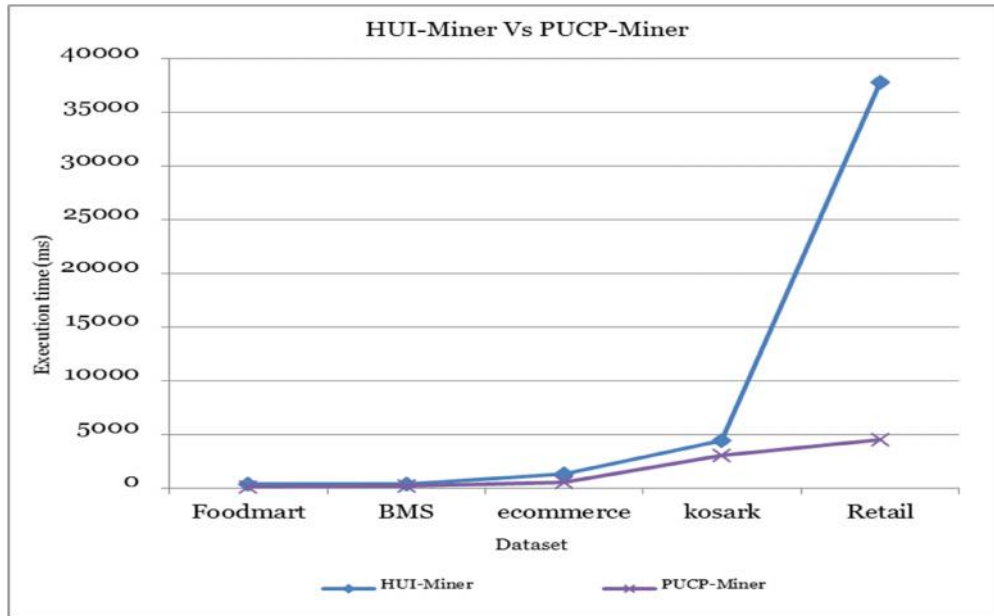


Figure 5.4: Execution time comparison HUI-Miner Vs PUCP-Miner.

Table 5.5: Execution time comparison mHUI-Miner Vs PUCP-Miner.

Dataset	mHUI-Miner (Execution time in ms)	PUCP-Miner (Execution time in ms)	Improvement (%)
Foodmart	161	131	18.63
BMS	402	218	45.77
ecommerce	1434	497	65.34
kosark	4030	3020	25.06
Retail	17002	4471	73.7

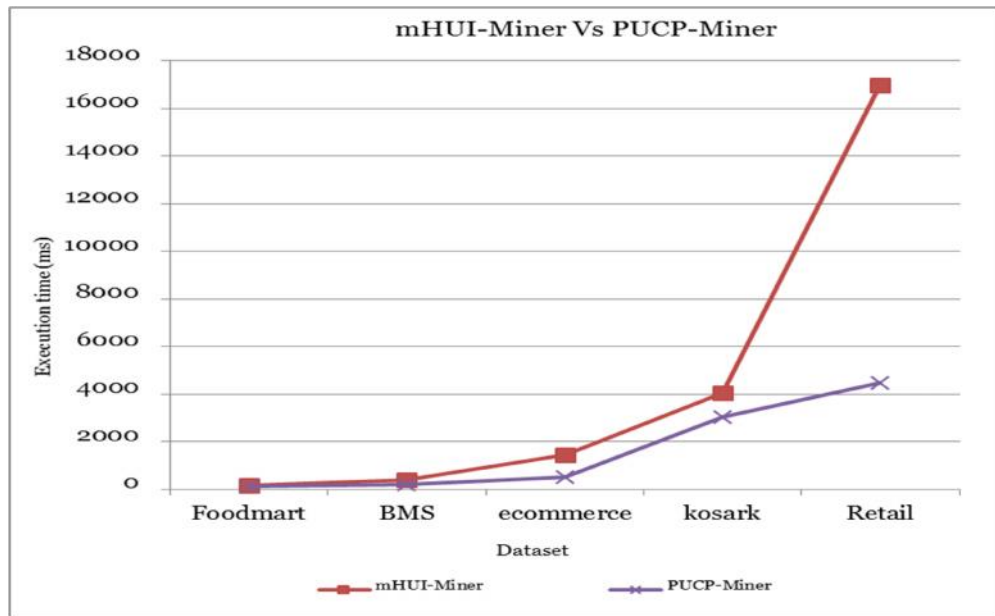


Figure 5.5: Execution time comparison mHUI-Miner Vs PUCP-Miner.

Table 5.6: Execution time comparison ULB-Miner Vs PUCP-Miner

Dataset	ULB-Miner (Execution time in ms)	PUCP-Miner (Execution time in ms)	Improvement (%)
Foodmart	160	131	18.13
BMS	378	218	42.33
ecommerce	624	497	20.35
kosark	4112	3020	26.56
Retail	6897	4471	35.17

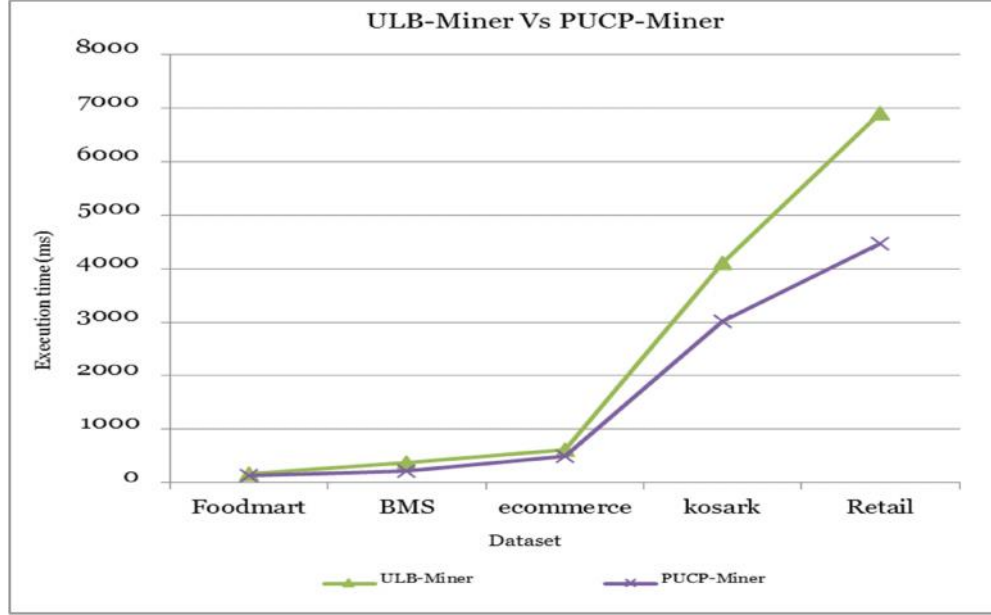


Figure 5.6: Execution time comparison HUI-Miner Vs PUCP-Miner

It has been observed from overall result analysis shown in Table 5.7 that using only SCAO based approach improves the performance of HUI-Miner by approximately 14 %, mHUI-Miner by 16%, and ULB-Miner by 17%. The approach incorporating both SCAO-based search space exploration and PUCP in PUCP-Miner is approximately 59% faster than HUI-Miner, approximately 46% faster than mHUI-Miner, and approximately 29% faster than ULB-Miner.

Table 5.7: overall improvement of proposed approaches

Algorithms	Average improvement (%)	
	SCAO Based Approach	PUCP-Miner (SCAO + PUCP)
HUI-Miner	15.00	58.71
mHUI-Miner	15.55	45.7
ULB-Miner	17.26	28.50

5.4.2 Memory Analysis

The memory requirements of PUCP-Miner and existing approaches mHUI-Miner and ULB-Miner on different datasets are listed in Tables 5.8 and 5.9, respectively. Memory requirement is also plotted in Figures 5.7 and 5.8 for performance comparison. The findings from the experiments indicate that PUCP-Miner utilizes significantly less memory compared

to mHUI-Miner and ULB-Miner on the ecommerce dataset, with reductions of approximately 46% and 8%, respectively. On the BMS dataset, PUCP-Miner consumes around 12%, and 13% less memory than mHUI-Miner, and ULB-Miner, respectively. When considering the Foodmart dataset, PUCP-Miner demonstrates a memory reduction of 47% and 8% compared to both mHUI-Miner and ULB-Miner. For the retail dataset, PUCP-Miner requires 32% and 12% less memory than mHUI-Miner and ULB-Miner, respectively. And finally, on the Kosark dataset, PUCP-Miner consumes approximately 9% and 7% less memory than mHUI-Miner and ULB-Miner.

Table 5.8: Memory Requirement (mHUI-Miner Vs PUCP-Miner)

Dataset	mHUI-Miner (memory in MB)	PUCP-Miner (memory in MB)	Improvement (%)
Foodmart	87.91	47.3	46.19
Retail	511.78	349.36	31.74
BMS	24.04	20.92	12.98
Ecommerce	87.91	47.3	46.19
Kosark	509	465	8.64

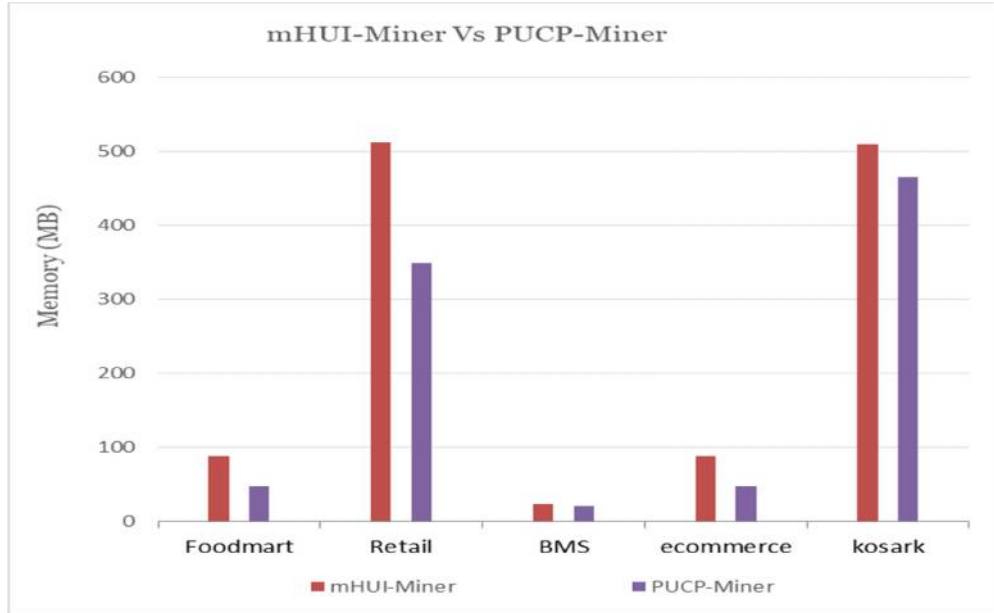


Figure 5.7: Memory comparison mHUI-Miner Vs PUCP-Miner

Table 5.9: Memory Requirement (ULB-Miner Vs PUCP-Miner)

Dataset	ULB-Miner (memory in MB)	PUCP-Miner (Memory in MB)	Improvement (%)
Foodmart	51.17	47.3	7.56
Retail	398.82	349.36	12.4
BMS	24.11	20.92	13.23
Ecommerce	51.17	47.3	7.56
Kosark	495.79	465	6.21

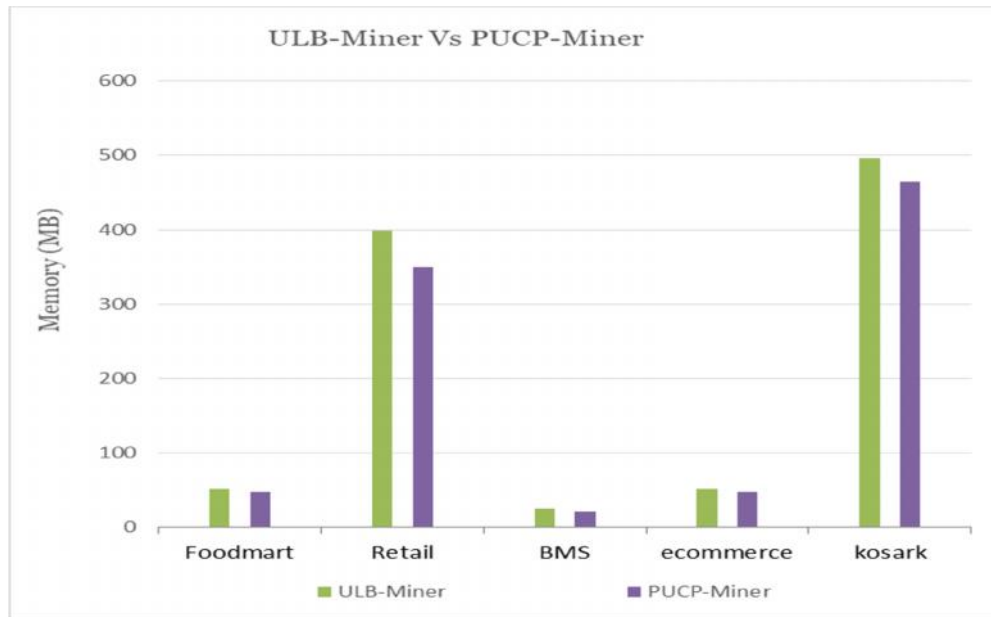


Figure 5.8: Memory comparison ULB-Miner Vs PUCP-Miner

CHAPTER-6

Conclusion and Future Work

6.1 Conclusion

Nowadays, every organization has to design an efficient decision support system to sustain their business in recent tough competition. The organization has vast amount of unused data of employees, customers, products, services, etc. The knowledge or information hidden inside the data can be helpful in designing a proper DSS. An association rule mining: a data mining technique that discovers the knowledge from the raw data. FIM is the conventional approach of association rule mining. The main limitation of the FIM is that it considers only the presence or absence of the items in the data. But in real-life applications with the item's presence, it is highly necessary to take the item's quantity and importance into account. HUIM (High Utility Itemset Mining) considers the item's quantity with its importance/weight/profit. The pattern derived from the HUIM is more significant than conventional FIM. In the last two decades, HUIM become an emerging trend in the research community. Among the most research works carried out for HUIM with various approaches, the utility list-based approaches are outstanding as it does not generate the candidate itemset. However, the performance of the utility list-based approaches are limited due to costly utility list join operations. These approaches also perform unnecessary utility list join operations for the low utility itemsets. The cost of the join operation is directly related to the number of comparisons required to search the common transactions between utility lists. To reduce the cost of the join operation, it is essential to minimize the number of comparisons required to search common transactions between the utility lists. SCAO-based search space exploration techniques greatly reduce such comparisons. Hence, it improved the performance of the HUIM approaches. PUCP (Predicted Utility Co-exist Pruning) uses the PUCS (Predicted Utility Co-exist Structure) to eliminate the unnecessary utility list join operations for the low utility itemsets. PUCP decides in advance whether it is necessary to perform join operations or discard the itemset. It reduces the number of join operations. Comprehensive experiments were conducted on a variety of real datasets. The experimental results show that SCAO-based search space exploration techniques improve the

performance of the HUIM approach from 13 to 18 percent. It improved the performance of the algorithm because the SCAO-based approach greatly reduced the number of comparisons from 12 to 27 percent. The combination of both SCAO-based search space exploration techniques and PUCP called PUCP-Miner has notable improvement in running time from 28 to 59 percent. It also consumes 8 to 46 percent less memory than other state-of-the-art methods on some standard real datasets.

6.2 Future Work

This thesis work mainly design with a focus on a static dataset. Since the database is changed often in real-world applications, more work will be needed to accommodate streams and dynamic datasets. Future work will involve adapting the model to handle streaming data and dynamic datasets where the database is continuously updated. This requires the development of algorithms that can efficiently process and analyse data in real-time.

Future work for this thesis can be directed towards several other key areas to enhance its applicability and robustness. The following aspects can be considered for further expansion: This thesis only considered the constant and positive utility value of the object. Reflecting the shifting preferences and importance of items over time requires careful evaluation of dynamic utility values. In order to enable the system to adjust to changing user preferences, future work will require creating processes to record and simulate the dynamics of utility values. A further level of complexity is added to the model by including negative utility values. It will function for the items in reference to the utility limitations in the future. As the dataset grows in size and complexity, scalability becomes crucial. Future work will focus on optimizing algorithms and techniques to handle large-scale datasets efficiently, ensuring that the model remains practical and scalable for real-world applications.

By addressing these aspects, the research can evolve to create a more sophisticated and versatile system capable of handling the challenges posed by real-world, dynamic datasets and diverse user preferences.

REFERENCES

- [1] Bishop, T., Reinke, J., & Adams, T. (2011). Globalization: Trends and Perspectives. *Journal of international business research*, 10, 117.
- [2] Rinaldi, M., Parretti, C., Salimbeni, L. B., & Citti, P. (2015). Conceptual design of a decision support system for the economic sustainability of nonprofit organizations. *Procedia CIRP*, 34, 119-124.
- [3] Sadiku, A. E. M., & Matthew, N. O. (2015). Shadare and SM, “A Brief Introduction to Data Mining,” *Eur. Sci. J*, 11(21), 1-3.
- [4] Lake, P., Crowther, P., Lake, P., & Crowther, P. (2013). Data, an organisational asset. *Concise Guide to Databases: A Practical Introduction*, 3-19.
- [5] Xu JJ. (2014). Knowledge discovery and data mining. in *Computing Handbook Information Systems and Information Technology*, Third Ed (pp. 1–22).
- [6] Maimon, O., Rokach, L. (2009). Introduction to Knowledge Discovery and Data Mining. In: Maimon, O., Rokach, L. (eds) *Data Mining and Knowledge Discovery Handbook*. Springer, Boston, MA. https://doi.org/10.1007/978-0-387-09823-4_1.
- [7] Agrawal, R., & Srikant, R. (1994, September). Fast algorithms for mining association rules. In *Proc. 20th int. conf. very large data bases, VLDB* (Vol. 1215, pp. 487-499).
- [8] Hokey Min (2006) Developing the profiles of supermarket customers through data mining, *The Service Industries Journal*, 26:7, 747-763, DOI: 0.1080/02642060600898252.
- [9] Luna, J. M., Fournier-Viger, P., & Ventura, S. (2019). Frequent itemset mining: A 25 years review. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 9(6), e1329.
- [10] Brauckhoff, D., Dimitropoulos, X., Wagner, A., & Salamatian, K. (2009, November). Anomaly extraction in backbone networks using association rules. In *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement* (pp. 28-34).
- [11] Yao, H., Hamilton, H. J., & Butz, C. J. (2004, April). A foundational approach to mining itemset utilities from databases. In *Proceedings of the 2004 SIAM International Conference on Data Mining* (pp. 482-486). Society for Industrial and Applied Mathematics.
- [12] Iqbal, M., Setiawan, M. N., Irawan, M. I., Khalif, K. M. N. K., Muhammad, N., &

- Aziz, M. K. B. M. (2022). Cardiovascular disease detection from high utility rare rule mining. *Artificial Intelligence in Medicine*, 131, 102347.
- [13] Pillai, J., & Vyas, O. P. (2010). Overview of itemset utility mining and its applications. *International Journal of Computer Applications*, 5(11), 9-13.
- [14] Zida, S., Fournier-Viger, P., Lin, J. C. W., Wu, C. W., & Tseng, V. S. (2017). EFIM: a fast and memory efficient algorithm for high-utility itemset mining. *Knowledge and Information Systems*, 51(2), 595-625.
- [15] Zhang, C., Almpandis, G., Wang, W., & Liu, C. (2018). An empirical evaluation of high utility itemset mining algorithms. *Expert Systems with applications*, 101, 91-115.
- [16] Ristovska, K., & Ristovska, A. (2014). The impact of globalization on the business. *Economic Analysis*, 47(3-4), 83-89.
- [17] Voleti S (2017). The Value of Data : A Motivating Example. *Essentials Bus. Anal.* p. 19–39.
- [18] Bavdaž, M., Snijkers, G., Sakshaug, J. W., Brand, T., Haraldsen, G., Kurban, B., ... & Willimack, D. K. (2020). Business data collection methodology: Current state and future outlook. *Statistical Journal of the IAOS*, 36(3), 741-756.
- [19] Chan, S. L., & Ip, W. H. (2011). A dynamic decision support system to predict the value of customer for new product development. *Decision support systems*, 52(1), 178-188.
- [20] Han, J., Kamber, M., & Pei, J. (2012). *Data mining concepts and techniques* third edition. University of Illinois at Urbana-Champaign Micheline Kamber Jian Pei Simon Fraser University.
- [21] Tan, P.N., Steinbach, M. and Kumar, V. (2016) *Introduction to Data Mining*. Pearson Education India, New Delhi.
- [22] Kesavaraj, G., & Sukumaran, D.S. (2013). A study on classification techniques in data mining. 2013 Fourth International Conference on Computing, Communications and Networking Technologies (ICCCNT), 1-7.
- [23] Pande, S., Sambare, S.S., & Thakre, V.M. (2012). *Data Clustering Using Data Mining Techniques*.
- [24] Abhinav Rai (2022). An Overview of Association Rule Mining & its Applications;5:927–30.
- [25] Fayyad, U., Piatetsky-Shapiro, G., & Smyth, P. (1996). From data mining to knowledge discovery in databases. *AI magazine*, 17(3), 37-37.

- [26] Gullo, F. (2015). From patterns in data to knowledge discovery: What data mining can do. *Physics Procedia*, 62, 18-22.
- [27] Borgelt, C. (2012). Frequent item set mining. *Wiley interdisciplinary reviews: data mining and knowledge discovery*, 2(6), 437-456.
- [28] Agrawal, R., & Srikant, R. (1994, September). Fast algorithms for mining association rules. In *Proc. 20th int. conf. very large data bases, VLDB* (Vol. 1215, pp. 487-499).
- [29] Grahne, G., & Zhu, J. (2005). Fast algorithms for frequent itemset mining using fp-trees. *IEEE transactions on knowledge and data engineering*, 17(10), 1347-1362.
- [30] Pyun, G., Yun, U., & Ryu, K. H. (2014). Efficient frequent pattern mining based on linear prefix tree. *Knowledge-Based Systems*, 55, 125-139.
- [31] Zhang, X. H., He, Y. D., Wan, J. H., & Zhao, H. (2001). An Improved Algorithm for Mining Association Rules. *JOURNAL-NORTHEASTERN UNIVERSITY NATURAL SCIENCE*, 22, 401-404.
- [32] Erwin, A., Gopalan, R. P., & Achuthan, N. R. (2007, October). CTU-Mine: An efficient high utility itemset mining algorithm using the pattern growth approach. In *7th IEEE international conference on computer and information technology (CIT 2007)* (pp. 71-76). IEEE.
- [33] Erwin, A., Gopalan, R. P., & Achuthan, N. R. (2008). Efficient mining of high utility itemsets from large datasets. In *Advances in Knowledge Discovery and Data Mining: 12th Pacific-Asia Conference, PAKDD 2008 Osaka, Japan, May 20-23, 2008 Proceedings 12* (pp. 554-561). Springer Berlin Heidelberg.
- [34] Han, J., Pei, J., & Yin, Y. (2000). Mining frequent patterns without candidate generation. *ACM sigmod record*, 29(2), 1-12.
- [35] Liu, Y., Liao, W. K., & Choudhary, A. (2005, August). A fast high utility itemsets mining algorithm. In *Proceedings of the 1st international workshop on Utility-based data mining* (pp. 90-99).
- [36] Coenen, F. (2011). Data mining: past, present and future. *The Knowledge Engineering Review*, 26(1), 25-29.
- [37] Gupta, M. K., & Chandra, P. (2020). A comprehensive survey of data mining. *International Journal of Information Technology*, 12(4), 1243-1257.
- [38] Jentner, W., Keim, D.A. (2019). Visualization and Visual Analytic Techniques for Patterns. In: Fournier-Viger, P., Lin, JW., Nkambou, R., Vo, B., Tseng, V. (eds) *High-Utility Pattern Mining. Studies in Big Data*, vol 51. Springer, Cham. https://doi.org/10.1007/978-3-030-04921-8_12.

- [39] Gan, W., Lin, J. C. W., Fournier-Viger, P., Chao, H. C., Tseng, V. S., & Philip, S. Y. (2019). A survey of utility-oriented pattern mining. *IEEE Transactions on Knowledge and Data Engineering*, 33(4), 1306-1327.
- [40] Nguyen, Loan TT, Phuc Nguyen, Trinh DD Nguyen, Bay Vo, Philippe Fournier-Viger, and Vincent S. Tseng. "Mining high-utility itemsets in dynamic profit databases." *Knowledge-Based Systems* 175 (2019): 130-144.
- [41] Shen, Y. D., Zhang, Z., & Yang, Q. (2002, December). Objective-oriented utility-based association mining. In *2002 IEEE International Conference on Data Mining, 2002. Proceedings.* (pp. 426-433).
- [42] R. Chan, Q. Yang and YDS. (2003) Mining high utility itemsets. *Third IEEE Int. Conf. Data Mining, 2003*, p. 19–26.
- [43] Yao H, Hamilton HJ. (2006). Mining itemset utilities from transaction databases. *Data Knowl Eng* 2006;59:603–26.
- [44] Liu, Y., Liao, W. K., & Choudhary, A. (2005). A two-phase algorithm for fast discovery of high utility itemsets. In *Advances in Knowledge Discovery and Data Mining: 9th Pacific-Asia Conference, PAKDD 2005, Hanoi, Vietnam, May 18-20, 2005. Proceedings 9* (pp. 689-695). Springer Berlin Heidelberg.
- [45] Li, Y. C., Yeh, J. S., & Chang, C. C. (2005). Direct candidates generation: a novel algorithm for discovering complete share-frequent itemsets. In *Fuzzy Systems and Knowledge Discovery: Second International Conference, FSKD 2005, Changsha, China, August 27-29, 2005, Proceedings, Part II 2* (pp. 551-560). Springer Berlin Heidelberg.
- [46] Li, Y. C., Yeh, J. S., & Chang, C. C. (2008). Isolated items discarding strategy for discovering high utility itemsets. *Data & Knowledge Engineering*, 64(1), 198-217.
- [47] Hu, J., & Mojsilovic, A. (2007). High-utility pattern mining: A method for discovery of high-utility item sets. *Pattern Recognition*, 40(11), 3317-3324.
- [48] Ahmed, C. F., Tanbeer, S. K., Jeong, B. S., & Lee, Y. K. (2009). An efficient candidate pruning technique for high utility pattern mining. In *Advances in Knowledge Discovery and Data Mining: 13th Pacific-Asia Conference, PAKDD 2009 Bangkok, Thailand, April 27-30, 2009 Proceedings 13* (pp. 749-756). Springer Berlin Heidelberg.
- [49] Ahmed, C. F., Tanbeer, S. K., Jeong, B. S., & Lee, Y. K. (2009). Efficient tree structures for high utility pattern mining in incremental databases. *IEEE Transactions on Knowledge and Data Engineering*, 21(12), 1708-1721.

- [50] Han, J., Pei, J., Yin, Y., & Mao, R. (2004). Mining frequent patterns without candidate generation: A frequent-pattern tree approach. *Data mining and knowledge discovery*, 8, 53-87.
- [51] Tseng, V. S., Wu, C. W., Shie, B. E., & Yu, P. S. (2010, July). UP-Growth: an efficient algorithm for high utility itemset mining. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 253-262).
- [52] Tseng, V. S., Shie, B. E., Wu, C. W., & Philip, S. Y. (2012). Efficient algorithms for mining high utility itemsets from transactional databases. *IEEE transactions on knowledge and data engineering*, 25(8), 1772-1786.
- [53] Ryang, H., Yun, U., & Ryu, K. H. (2016). Fast algorithm for high utility pattern mining with the sum of item quantities. *Intelligent Data Analysis*, 20(2), 395-415.
- [54] Liu, M., & Qu, J. (2012, October). Mining high utility itemsets without candidate generation. In *Proceedings of the 21st ACM international conference on Information and knowledge management* (pp. 55-64).
- [55] Peng, A. Y., Koh, Y. S., & Riddle, P. (2017). mHUIMiner: A fast high utility itemset mining algorithm for sparse datasets. In *Advances in Knowledge Discovery and Data Mining: 21st Pacific-Asia Conference, PAKDD 2017, Jeju, South Korea, May 23-26, 2017, Proceedings, Part II 21* (pp. 196-207). Springer International Publishing.
- [56] Fournier-Viger, P., Wu, C. W., Zida, S., & Tseng, V. S. (2014). FHM: Faster high-utility itemset mining using estimated utility co-occurrence pruning. In *Foundations of Intelligent Systems: 21st International Symposium, ISMIS 2014, Roskilde, Denmark, June 25-27, 2014. Proceedings 21* (pp. 83-92). Springer International Publishing.
- [57] Duong, Q. H., Fournier-Viger, P., Ramampiaro, H., Nørnvåg, K., & Dam, T. L. (2018). Efficient high utility itemset mining using buffered utility-lists. *Applied Intelligence*, 48, 1859-1877.
- [58] Qu, J. F., Liu, M., & Fournier-Viger, P. (2019). Efficient algorithms for high utility itemset mining without candidate generation. *High-Utility Pattern Mining: Theory, Algorithms and Applications*, 131-160.
- [59] Wu, P., Niu, X., Fournier-Viger, P., Huang, C., & Wang, B. (2022). UBP-Miner: An efficient bit based high utility itemset mining algorithm. *Knowledge-Based Systems*, 248, 108865.

- [60] Fournier-Viger, P., Lin, C.W., Gomariz, A., Gueniche, T., Soltani, A., Deng, Z., Lam, H. T. (2016). The SPMF Open-Source Data Mining Library Version 2. Proc. 19th European Conference on Principles of Data Mining and Knowledge Discovery (PKDD 2016) Part III, Springer LNCS 9853, pp. 36-40

List of Publications

1. Patel Suresh B, Sanjay M. Shah, and Mahendra N. Patel "An Efficient High Utility Itemset Mining Approach using Predicted Utility Co-exist Pruning." International Journal of Intelligent Systems and Applications in Engineering 10, no. 4 (2022): 224-230. **(SCOPUS Approved, ISSN: 2147-6799)**
2. A Patel Suresh B, Sanjay M. Shah, and Mahendra N. Patel "An Efficient Search Space Exploration Technique for High Utility Itemset Mining." Procedia Computer Science 218 (2023): 937-948. **(SCOPUS Approved, ISSN: 1877 0509)**

