UTILITY BASED AND USER DEFINED SCORING BASED MINING OF SEQUENTIAL PATTERNS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

ÖZNUR KIRMEMIŞ ALKAN

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF DOCTOR OF PHILOSOPHY
IN
COMPUTER ENGINEERING

MAY 2015

Approval of the thesis:

UTILITY BASED AND USER DEFINED SCORING BASED MINING OF SEQUENTIAL PATTERNS

submitted by **ÖZNUR KIRMEMIŞ ALKAN** in partial fulfillment of the requirements for the degree of **Doctor of Philosophy in Computer Engineering Department, Middle East Technical University** by,

Prof. Dr. Gülbin Dural Ünver _____
Dean, Graduate School of **Natural and Applied Sciences**

Prof. Dr. Adnan Yazıcı _____
Head of Department, **Computer Engineering**

Assoc. Prof. Dr. Pınar Karagöz _____
Supervisor, **Computer Engineering Department, METU**

Examining Committee Members:

Prof. Dr. Özgür Ulusoy
Computer Engineering Department, Bilkent University _____

Assoc. Prof. Dr. Pınar Karagöz
Computer Engineering Department, METU _____

Prof. Dr. İsmail Hakkı Toroslu
Computer Engineering Department, METU _____

Prof. Dr. Nihan Kesim Çiçekli
Computer Engineering Department, METU _____

Assoc. Prof. Dr. Osman Abul
Computer Engineering Department, TOBB University _____

Date: _____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name: ÖZNUR KIRMEMIŞ ALKAN

Signature :

ABSTRACT

UTILITY BASED AND USER DEFINED SCORING BASED MINING OF SEQUENTIAL PATTERNS

Alkan, Öznur Kırmemiş

Ph.D., Department of Computer Engineering

Supervisor : Assoc. Prof. Dr. Pınar Karagöz

May 2015, 103 pages

Sequential pattern mining is an important data mining problem with broad applications. The classical frequency-based solutions often lead to many patterns being identified, most of which are not informative for the end-users. To handle this problem, utility based mining technique emerged, which assign non-binary values, called utilities, to items and calculate pattern utilities accordingly. In the thesis work, two new frameworks are proposed in response to the challenges and limitations of the existing solutions in utility based sequence mining. The first solution is a new framework for high utility sequential pattern mining, which presents efficient data structures and a new pruning technique that is based on Cumulated Rest of Match (CRoM) based upper bound so as to efficiently prune the huge combinatorial search space. CRoM, by defining a tighter upper bound on the utility of the candidates, allows more conservative pruning before candidate pattern generation in comparison to the existing techniques. In addition, an efficient algorithm, HuspExt (High Utility Sequential Pattern Extraction), have been developed, which calculates the utilities of the child patterns based on that of the parents'. Substantial experiments on both synthetic and real datasets from different domains show that, the proposed solution efficiently discovers high utility sequential patterns from large scale datasets with different data characteristics, under low utility thresholds. The second solution presents a new approach for sequential pattern extraction for the cases where utility definition is not adequate to

define the value of the patterns. This solution is based on user-defined scoring mechanism, and the proposed solution is evaluated under the web usage domain. Evaluation of this solution on real datasets from web domain prove that, the solution effectively discovers patterns under user defined scoring mechanism.

Keywords: Sequential Pattern Mining, High Utility Sequential Pattern Mining, Efficiency, Candidate Pattern Pruning, Web Usage Mining, Web Access Sequence, Web Access Pattern, User Defined Pattern Scoring

ÖZ

FAYDAYA BAĞLI VE KULLANICI TANIMLI SKORLAMAYA BAĞLI SIRALI DESEN MADENCİLİĞİ

Alkan, Öznur Kırmemiş
Doktora, Bilgisayar Mühendisliği Bölümü
Tez Yöneticisi : Doç. Dr. Pınar Karagöz

Mayıs 2015 , 103 sayfa

Sıralı desen çıkarımı, geniş uygulamaları olan bir veri madenciliği problemidir. Klasik, sıklığa dayalı çözümler, çoğunlukla son kullanıcı için bilgi verici olmayan çok fazla sayıda desen bulunmasına yol açmaktadır. Bu problemi çözmek için, ikili olmayan, fayda denilen değerleri nesnelere atayan, faydaya dayalı çıkarım teknikleri ortaya çıkmıştır. Bu tez çalışmasında, faydaya dayalı sıralı desen çıkarımı için önerilen çözümlerin varolan eksikliklerine ve zorluklarına cevaben iki yeni çatı geliştirilmiştir. İlk çözüm, yüksek faydaya dayalı sıralı desen çıkarımı için, verimli veri yapıları, ve büyük arama alanını budamak için, CRoM (Birikmiş Kalan Uyum)'a bağlı üst limiti kullanarak yeni bir budama tekniği sunan bir çatıdır. CRoM, aday desenlerin faydaları üzerinde daha sıkı bir üst limit tanımlayarak, varolan tekniklere kıyasla daha ölçülü bir budama sağlamaktadır. Buna ek olarak, HuspExt (Yüksek Faydaya Dayalı Sıralı Desen Çıkarımı) adlı çocuk desenlerin faydasını ana desenden hesaplayan verimli bir algoritma geliştirilmiştir. Farklı alanlara ait, hem sentetik hem de gerçek veri kümeleri üzerinde yapılan deneyler göstermektedir ki, önerilen yaklaşım yüksek faydaya dayalı sıralı desenleri, farklı özelliklerdeki büyük veri kümelerinden, düşük fayda limitlerinde dahi etkili bir şekilde çıkarmaktadır. İkinci çözüm, fayda tanımının, desenlerin değerlerini tanımlamada yeterli olmadığı durumlar için yeni bir yaklaşım sunmaktadır. Bu çözüm, kullanıcı tanımlı skorlama mekanizmasına bağlıdır ve şu anki versiyonu web kullanımı alanında değerlendirilmiştir. Gerçek veriler üze-

rinde yapılan deneyler göstermektedir ki, ikinci çözüm, kullanıcı tanımlı skorlama mekanizması altında desenleri etkin bir şekilde çıkarmaktadır.

Anahtar Kelimeler: Sıralı Desen Çıkarımı, Yüksek Faydaya Bağlı Sıralı Desen Çıkarımı, Etkinlik, Aday Desen Budaması, Web Kullanım Madenciliği, Web Erişim Dizgesi, Web Erişim Deseni, Kullanım Tanımı Desen Skorlama

To my husband Ozan, my twins Masal and Uras

ACKNOWLEDGMENTS

I am greatly indebted to many people without whom this study might not come to an end.

Foremost, I am grateful to my supervisor Dr. Pınar Karagöz for believing in me, guiding me and motivating me through the process. She is more than a supervisor to me; she is my guide and friend.

I am also thankful to my thesis committee members, Prof. Dr. Özgür Ulusoy and Prof. Dr. İsmail Hakkı Toroslu, for their motivation and valuable comments throughout this study.

I take this opportunity to record, my sincere gratitude to Scientific and Technological Research Council of Turkey (TÜBİTAK) for providing me Ph.D. fellowship (2211).

Last but not the least, I would like to thank my family, my mother Kevser Kırmemiş, my father Davut Kırmemiş, my sister Özlem Kırmemiş Yüce for being a part of my life. I would like to thank my aunt Çakır Durmuş for her endless support during my PhD study. My special thanks is to my husband Ozan Alkan, and my lovely twins Masal Alkan and Uras Alkan for supporting me unconditionally throughout my study.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

LIST OF ABBREVIATIONS

PBCG	Pruning Before Candidate Generation
PACG	Pruning After Candidate Generation
CRoM	Cumulated Rest of Match
HuspExt	High Utility Sequential Pattern Extraction
BF	Branching Factor
KDD	Knowledge Discovery in Database
DCP	Downward Closure Property
TWU	Transaction Weighted Utilization
GSP	Generalized Sequential Pattern
SPADE	Sequential PAttern Discovery using Equivalent classes
SPAM	Sequential Pattern Mining
FP	Frequent Pattern
FreeSpan	Frequent Pattern Projected Sequential Pattern Mining
WAP	Web Access Pattern
PTAC	Sequential Frequent Patterns mining with Tough Aggregate Constraints
GTC	Graph for Time Constraints
SPIRIT	Sequential Pattern Mining with Regular Expression Constraints
TWDC	Transaction-Weighted Downward Closure
SWU	Sequence-Weighted Utility
UL	UtilityLevel
UL	UtilitySpan
SUUB	Sequence Utility Upper Bound
SDCP	Sequence Weighted Downward Closure Property
WaPUPS	Web Access Pattern Extraction Under User Defined Pattern Scoring
URL	Uniform Resource Identifier
PACG	Pruning After Candidate Generation
RMUB	Rest of Match Based Upper Bound

CSeq	Containing Sequences

CHAPTER 1

INTRODUCTION

1.1 Motivation

With the progress of development of technology, there has been an increase in our capabilities for both generating and collecting data. In order to utilize this massive data effectively, it is important to extract both implicit and explicit unidentified patterns which can direct the process of decision making. *Data mining* is the area of research that includes techniques and principles to extract patterns that describe the characteristic properties of this huge data [22]. *Frequent pattern mining* is one of the major fields of data mining, which aims to discover patterns that appear in a dataset with frequency no less than a user-specified minimum support threshold.

Frequent pattern mining has been studied extensively by data mining researchers [2, 46, 12, 31, 1] and it is used in a wide spectrum of areas, such as financial data analysis, retail industry, customer shopping history, goods transportation, consumption and services, telecommunication industry, biological data analysis, scientific applications, and network intrusion detection.

Although frequent pattern mining has been used in many applications, end-users may not be interested in patterns that occur at a high ratio in datasets. In other words, support may not always define the interestingness of the extracted patterns. In many real-life applications, rare patterns can provide useful information in different decision-making domains such as business transactions, medical, security, fraudulent transactions, and retail communities. For example, in a supermarket, customers purchase microwave ovens or teapots rarely as compared to sugar, fruits or bread. But for the

supermarket, the former transactions yield more profit. Similarly, the high-profit rare patterns are found to be very useful in many application areas. For example, the rare combination of symptoms can provide useful insights for doctors in medical applications. The classical frequency-based frameworks cannot solve such problems and this leads to the emergence of *high utility pattern mining* [59].

In high utility pattern mining, the meaning of *utility* refers to the interestingness, importance, or profitability of the patterns, and the aim is to extract patterns with utility greater than or equal to a *minimum utility threshold*. The success of any high utility sequential pattern mining solution depends on its power to restrict the number of the candidates and simplify the computation for calculating the utility [36]. A tighter upper bound on the utility of candidate patterns may result in pruning the search space better. In addition, the phase of the solutions at which the pruning mechanism is used, effects the performance of the algorithms. Pruning the search space can be performed either prior to or after candidate generation, which are referred to as *PBCG (Pruning Before Candidate Generation)* and *PACG (Pruning After Candidate Generation)*, respectively. In comparison to PACG, PBCG saves more space and time, since candidates are eliminated before they are generated and tested. Moreover, most of the computational complexity of pattern extraction lies in the calculation of the utilities, and to lower this complexity, efficient data structures should be utilized.

Although utility-based sequential pattern mining provides a solution for the limitations of frequent sequential pattern mining, it does not cover all real-life scenarios. In other words, existing utility-based pattern mining techniques calculate pattern utilities based on the utility of the individual items. However, a pattern's value cannot always be calculated from the distinct item utilities. For instance, a user may be interested in patterns that exist in at least for a number of distinct transactions in the dataset. Similarly, for the case of discovering interesting patterns from web log data, for instance, the value of a pattern may depend on the average income of the users that traverse the pattern. In utility-based frameworks, such values cannot be calculated from the web page utilities. Instead, they should be calculated for each extracted pattern on its own. In addition, common to both frequency-based and utility-based techniques, one important challenge is that, there exists a huge number of possible patterns that are hidden in databases and the mining algorithm should find the complete set of patterns,

when possible, satisfying the minimum support or the minimum utility threshold. For especially huge and sparse datasets, it is not possible to extract patterns under high threshold values. For low threshold values, however, state of the art algorithms may fail to respond.

The contribution of this thesis study to literature is two-fold. First, we have developed a generic framework for high utility sequential pattern mining, which introduces a tight upper bound, *CRoM (Cumulated Rest of Match)* based upper bound that is used for eliminating candidate patterns before generation, and *HuspExt (High Utility Sequential Pattern Extraction)* algorithm, that utilizes efficient data structures during utility calculations. We conduct a series of experiments both with synthetic and real datasets for examining the performance of this solution. The results show that, the proposed solution efficiently extracts high utility sequential patterns from large scale datasets under low utility thresholds.

Second, motivated by the limitations of the existing utility-based and frequency-based techniques, we present a new solution for extracting sequential patterns under user-defined pattern scoring. There exists two motivations behind developing this technique. First, we aim to provide a solution for the cases where the utility definition is not enough to define the value of a pattern. Second, we aim to control the size of the search space by defining a new parameter, *BF (Branching Factor)*, which enables the solution to extract high-valued patterns even in the cases for which existing frequency-based techniques and utility-based techniques cannot generate any pattern. The proposed solution can be used in any domain where the data under consideration is sequential. For the evaluation purposes, the current version of the solution is adapted to work with the web usage data. Therefore, the data under consideration is composed of sessions of users, where each session has a number of accesses corresponds to the web page requests, the users send to the server. The extracted patterns are web usage patterns that reveal valuable information about users' navigation behaviors.

The proposed framework is hybrid in the sense that it combines clustering with a new pattern extraction algorithm. Pattern extraction process uses an evaluation function that can be defined according to the preferences or needs of the users. The role of

clustering in the framework is as follows; considering different behavioral patterns, it may not be very useful to construct access paths directly from all the sessions, resulting in a one general and cumulated profile of the users. Therefore, the solution first discovers different navigation behaviors through clustering user sessions. After related session groups in terms of the navigation behavior are identified, the system utilizes the search-based pattern extraction algorithm so as to construct access patterns. The solution can be used in any problem domain where the resulting web access patterns can be utilized, such as adaptive web, web page recommendation, next page prediction, ad recommendation, and user-behavior analysis. In addition, it can be applied to any other sequence data, as well. Similar to our first solution, we perform a detailed experimental evaluation of our second solution under well-known datasets from literature. Evaluation phase includes experiments for determining the efficiency of the technique under different values of the system parameters and the comparison of the proposed framework with a well-known sequential pattern mining algorithm, PrefixSpan [46].

1.2 Contributions

The primary goal in this thesis is to propose solutions in response to limitations and challenges of the existing sequential pattern mining approaches. For this aim, we developed two solutions. In our first solution, we propose efficient data structures, a tight upper bound for candidate elimination before generation and a new algorithm, HuspExt, for high utility sequential pattern mining. In our second solution, we present a hybrid framework for user evaluation score based extraction of patterns and this solution is applied to web usage data. The contributions of the thesis, therefore, can be grouped into two parts, where each part corresponds to one of the proposed solutions.

The contributions of the first solution can be summarized as follows:

1. Important concepts and components of high utility sequential pattern mining problem are formalized. Especially, we formally introduce the *ordering* concept which is important for high utility sequential pattern mining problem, since

it is very effective on the performance of the solution.

2. A PBCG strategy is proposed which is based on CRoM. With this pruning strategy, a tight upper bound on the utility of the candidate patterns can be defined.

3. The algorithm, HuspExt, is proposed in order to lower the time and space requirements of the solution. Different from the existing techniques, a new data structure for the nodes of the pattern tree is used in order to apply CRoM based pruning.

4. Both real and synthetic datasets are used in the evaluation phase. Real datasets are from different domains and synthetic datasets have different characteristics so as to make a comprehensive evaluation of the solution. The performance of the proposed technique with the state-of-the-art high utility sequential pattern mining algorithms are compared.

The contributions of the second solution can be summarized as follows:

1. A hybrid approach is proposed for extracting web access patterns which combines clustering with a new pattern extraction algorithm. Clustering enables the solution to reduce the search space, and as shown by the experiments, it increases the accuracy of the solution through focusing on more relevant session groups.

2. Proposed pattern extraction algorithm utilizes a user-defined evaluation function which reflects the preferences of users in terms of the properties of the extracted patterns. Pattern values are calculated from any data provided to the solution, and different from existing utility-based techniques, this evaluation does not necessarily depend on individual web page utilities.

3. Proposed solution introduces the BF parameter, which gives a control over the size of the search space and enables the solution to generate high valued patterns even for cases where the state of art techniques cannot respond.

4. We demonstrate the proposed approach on an example user-defined path evaluation function. The evaluation function examined in the experiments is ap-

plicable in web domain and it produced useful results for facilitating user's navigation as shown by the experiments.

5. Comprehensive evaluation of the solution is performed on four real datasets. The performance of the proposed framework is compared with a well-known sequential pattern mining algorithm from the literature and the results are discussed.

1.3 Organization of the Thesis

This thesis work presents two solutions to sequential pattern mining problem from two different perspectives. The first approach presents a solution to utility-based sequential pattern mining. The second approach, on the other hand, is a new framework for user defined scoring based sequential pattern extraction, which is applied to web log data.

Chapter 2 covers our understanding of mining of sequential patterns and web usage mining. It presents the definitions of the problems that are explored in this thesis work, and it discusses the main challenges of the areas of research. Further the mechanisms to handle these limitations are introduced that also forms the problem definition. In addition, the chapter discusses the importance of value based sequential pattern mining.

Chapter 3 covers the comprehensive literature survey of algorithms for sequential pattern mining and web usage mining. Different approaches to sequential pattern mining including, frequency-based sequence mining, constraint-based techniques and utility-based solutions have been studied for their characteristics. Limitations and main challenges of these algorithms have been discussed. In addition, existing solutions to web usage mining are presented. This detailed survey has identified the limitations and main problems of the existing algorithms, and mentions the contributions of the proposed solutions in this thesis work, to the literature.

Chapter 4 describes the proposed solution to utility-based sequential pattern mining problem, which is a generic framework for high utility sequential pattern extraction.

The solution introduces a tight upper bound, *CRoM* based upper bound, that is used for eliminating candidate patterns before generation and *HuspExt* algorithm that utilizes efficient data structures during utility calculations.

Chapter 5 describes a hybrid framework which combines clustering with a new pattern extraction algorithm. The pattern extraction process uses an evaluation function that can be defined according to the preferences or needs of the users. The solution presents a new technique to overcome the limitations of existing sequential pattern mining solutions by both proposing a novel value based pattern extraction framework and a new parameter, BF, for controlling the huge search space of sequential pattern mining problem for especially big and sparse datasets.

Chapter 6 presents the detailed experimental evaluation of the proposed techniques. The datasets that are used in the experiments, the evaluation methodology and metrics conducted during experimentation, and the results of the evaluation phase are presented. Discussion of the results and comparison with the existing state of art techniques are provided in this chapter.

Chapter 7 concludes this thesis by providing a brief summary of what has been done in the thesis work. The limitations of this study together with the future research directions are also suggested in this chapter.

CHAPTER 2

BACKGROUND

There is a huge growth of data sources in the world every day and together with this growth, it becomes vital to process and analyze data, extract the important knowledge from it, understand it, and use it for further purposes. Statistical methods have been used for analysis of data for a long time; however, they are applicable for simple applications. There is a requirement for advanced techniques in order to perform intelligent data analysis to provide competitive advantage. *Data Mining* or *Knowledge Discovery* has been emerged as a research field in order to provide solutions in this area [22].

Data mining can be defined as *the computational process of discovering patterns in large datasets* that uses methods from different disciplines including *artificial intelligence, machine learning, statistics,* and *database systems*. The motivation behind data mining research is to extract implicit, previously unknown and potentially useful information from data through utilizing automatic or semi-automatic techniques in order to discover meaningful patterns. The overall goal of the data mining process is to extract information from a dataset and transform it into an understandable structure for further use. The extracted information that is of interest to the target user is referred as *knowledge*. A *pattern*, on the other hand, is defined as a collection of records in the database that have same attribute values. Both empirical and knowledge based techniques are required in order to extract useful patterns from huge and complex databases.

Generally speaking, there are two classes of data mining tasks, namely, *descriptive* and *prescriptive* [15]. *Descriptive mining* summarizes or characterizes the general

properties of data, while *prescriptive mining* performs inference on the current data to make predictions. Among the various descriptive data mining processes, *frequent pattern mining* is one of the most commonly employed task to facilitate business development, which is detailed in the next subsection.

2.1 Frequent Pattern Mining

Mining frequent set of items that occur together in transaction databases is a fundamental task for several forms of knowledge discovery such as association rules, sequential patterns, and classification [29]. Here, transactions are in the form of a set of items and recent research has focused on determining which *groups of items*, called *itemsets*, frequently appear together in transactions.

In order to perform this, *support* measure is utilized, which is used to determine the importance of an itemset in the transaction database. It is calculated as the percentage of all transactions that contain the itemset [29]. Using this measure, an itemset is determined as *frequent*, if its support is not less than a given threshold value. This threshold is called the *minimum support threshold*. Frequent itemset mining is generally adopted to generate association rules and applied heavily to market-basket analysis.

The classical framework of frequent pattern mining solutions generally uses the minimum support framework to discover complete set of frequent patterns. Minimum support threshold controls the minimum number of transactions a pattern must cover in a database, and the solutions in this area uses this information to prune the huge search space. More specifically, they are based on the *DCP (Downward Closure Property)* [2], which states that *all non-empty subsets of a frequent pattern must also be frequent*. Therefore, DCP plays a fundamental role for varieties of frequent pattern mining algorithms, since it holds the key for effectively pruning the search space.

2.2 Sequential Pattern Mining

Sequential pattern mining is similar to frequent pattern mining, with an important difference of having the events being linked by time [15]. Sequential patterns indicate the correlation between transactions, while association rule represents intra transaction relationships [54]. In other words, the extracted patterns from association rule mining are based on items that are purchased together frequently and that belong to same transaction. Patterns extracted by sequential pattern mining, on the other hand, are based on items that are brought in a certain order by the same customer; however, they can be from different transactions. This thesis work presents two new approaches to sequential pattern mining from two different perspectives.

Sequential pattern mining problem was first addressed by Agrawal and Srikant [54] and was defined as follows: *"Given a database of sequences, where each sequence consists of a list of transactions ordered by transaction time and each transaction is a set of items, sequential pattern mining is to discover all sequential patterns with a user-specified minimum support, where the support of a pattern is the number of data-sequences that contain the pattern."*

2.2.1 Definitions

Before formally defining the problem of mining sequential patterns, important definitions related to this research are given below.

Any sequential pattern mining framework is built upon the following components; *item, itemset, sequence* and *sequence database*. Let $I = \{i_1, i_2, ..., i_m\}$ be a finite set of distinct items and $SD = \{S_1, S_2, ..., S_n\}$ be a sequence database of *n* sequences.

Definition 2.2.1. *An itemset is defined as a nonempty unordered collection of items.*

Items in an itemset can be listed in any order, and without loss of generality, we assume that they are listed alphabetically.

Definition 2.2.2. *A sequence $S \in SD$, is defined as an ordered list of itemsets.*

Definition 2.2.3. *The size of an itemset is defined as the number of the items in it.*

For the exemplification of the ideas, a sequence database is provided in Table 2.1. For this example database, size of the first and second itemsets of S_1 are *1* and *2*, respectively.

Definition 2.2.4. *The size of a sequence is the number of the itemsets in that sequence.*

For instance, for the example database, size of the sequences S_1, S_2 and S_3 are *4*, *5*, and *2*, respectively.

Definition 2.2.5. *The length of a sequence is defined as the number of items in that sequence.*

For instance, length of the sequences S_1, S_2 and S_3 are *5*, *6*, and *3*, respectively.

Definition 2.2.6. *Given itemsets* $IS_a = \{i_{a_1}, i_{a_2}, ..., i_{a_n}\}$ *and* $IS_b = \{i_{b_1}, i_{b_2}, ..., i_{b_m}\}$*;* IS_a *contains* IS_b *if and only if there exists positive integers k, j, z such that; for* $n \geq m$, $k \geq 1$, $j > k$, $z > j$, $j, z \leq n$, $i_{b_1} = i_{a_k}$, $i_{b_2} = i_{a_j}$,..,$i_{b_m} = i_{a_z}$.

For instance, itemset (a, c) contains itemsets (a), and (a, c), but it does not contain itemsets (e) and (a, e).

Definition 2.2.7. *Given two sequences* $S_a =< IS_{a_1}, IS_{a_2}, ..., IS_{a_n} >$, $S_b =< IS_{b_1}, IS_{b_2}, ..., IS_{b_m} >$*;* S_a *contains* S_b *if and only if there exists positive integers i, j, z such that for* $n \geq m$, $i \geq 1$, $j > i$, $z > j$, $j, z \leq n$, IS_{a_i} *contains* IS_{b_1}, IS_{a_j} *contains* IS_{b_2},....., IS_{a_z} *contains* IS_{b_m}.

For instance, sequence $< (h)(c, d)(b) >$ contains the sequence $< (h, d) >$ but does not contain the sequence $< (h, d)(a) >$.

We discriminate between the terms *sequence* and *pattern* so as to clarify the definitions. In the paper, *sequence* refers to the database transactions and *pattern* refers to the extracted subsequences from these transactions. For instance, the database provided in Table 2.1 has three sequences, and $< (c)(d) >$ is a pattern that exists in sequences S_2 and S_3.

Table 2.1: Sequence Database

SID	Sequence
S_1	$< (d)(c,e)(b)(a) >$
S_2	$< (b)(c,d)(b)(d)(e)] >$
S_3	$< (b,c)(d) >$

2.2.2 Formalization of the Sequential Pattern Mining Problem

Considering the concepts defined above, the problem of sequential pattern mining can be formally defined as follows: *Given a sequence database SD, the support or frequency of a pattern is defined as the number of sequences in SD that contain this pattern. The task of any sequential pattern mining solution is to find all frequent patterns from SD whose support is greater than or equal to a user-supplied minimum support threshold*. This constitutes the problem definition for all sequence mining algorithms whose data are located in a transaction database or in transaction datasets [45].

2.3 Utility-Based Sequential Pattern Mining

Frequent pattern mining solutions, including sequential pattern mining techniques, does not reflect the impact of any other factor except frequency, that is, the presence or the absence of an item or itemset. However, for some cases, frequency may only contribute a small portion of the overall profit, and non-frequent items and itemsets may contribute a large portion of the profit. For instance, a retail business may be interested in identifying its customers who buy full priced items, high margin items, or gourmet items, which may be absent from a large number of transactions, because most customers do not buy these items. In a traditional frequency based approach, these transactions may be left out due to their low support values. For instance, *{milk, bread}* may be a frequent itemset with support higher than the minimum support threshold. However, such an itemset may contribute a very low portion of the total profit. *{birthday cake, birthday card}*, on the other hand, may be a non-frequent itemset with a low support value that is lower than the threshold; however, marketing

professionals must be more interested in promoting such itemsets. These itemsets are missed by a classical frequency based mining technique [37].

Another example can be given from the web mining domain where the sequence of web pages visited by a user constitutes a sequence database, and user sessions constitute the itemsets. Since the number of visits to a web page and the time spent on that web page differs for different users, the total time spent on a web page may determine the utility of the items. Based on this formulation, the website designers can catch the interests of the customers by looking at the utilities of the different page combinations, and therefore may consider re-organizing the link structure of the website to better suit to the preference of the users. For such a real-life scenario, frequency is not sufficient to provide a solution. In other words, support may not always define the interestingness of the patterns. Target users may even prefer to work on high-profit rare patterns, which may be more useful than frequent patterns for their applications. This necessity leads to the emergence of *high utility pattern mining* [59].

In high utility pattern mining, the meaning of *utility* refers to the interestingness, importance, or profitability of the patterns, and the aim is to extract patterns with utility greater than or equal to a *minimum utility threshold*. High utility itemset mining is the utility-based counterpart of frequent itemset mining, where the database transactions are itemsets, not the sequences of itemsets, as it is the case for utility based sequential pattern mining. In the following subsections, first, the important definitions related to this research are given. Next, the challenges of utility based sequential pattern mining research are presented.

2.3.1 Definitions

In this section, we describe the main terms of high utility sequential pattern mining and give formal definitions of important concepts, some of which have not been formalized before. We used the same utility function definitions as in the literature for the sake of compatibility and comparability with the existing techniques [59]. In addition, some terms are defined in the same way as they are defined in frequent sequential pattern mining framework. These definitions are repeated in this section also, but they

Table 2.2: Sequence Database

SID	Sequence
S_1	$< (d,1); [(c,1)(e,3)]; (b,7); (a,2) >$
S_2	$< (b,2); [(c,1)(d,3)]; [(b,1)(d,2)(e,4)] >$
S_3	$< [(a,2)(c,4)]; (b,3) >$
S_4	$< (b,1); (b,6); (a,1); (a,2); (c,3) >$
S_5	$< [(a,1)(c,2)]; (b,2); (d,1); (b,1) >$
S_6	$< [(b,4)(c,3)]; (d,2) >$

Table 2.3: External Utility Table

item	a	b	c	d	e
eu	2	2	4	1	3

are not separated in distinct definition headings. However, the definitions that differ in utility based sequence mining framework in comparison to frequent pattern mining are given below under distinct definition headings.

Similar to frequent sequential pattern mining framework, the framework of utility based sequential pattern mining is built upon the following components; *item, itemset, sequence, sequence database*, and *pattern*. Let $I = \{i_1, i_2, ..., i_m\}$ be a finite set of distinct items and $SD = \{S_1, S_2, ..., S_n\}$ be a sequence database of *n* sequences. Each *sequence* $S \in SD$, is an ordered list of itemsets, where each itemset is an unordered list of items. An *itemset* is formally represented as $IS = \{i_1, i_2, ..., i_p\}$, and defined as a set of *p* distinct items, where $i_j \in I$ for $1 \leq j \leq p$. Itemsets are surrounded with brackets if it has more than one item, however, for brevity, they are omitted if it has one item. Items in an itemset can be listed in any order, and without loss of generality, we assume that they are listed alphabetically. Each *sequence S* in the sequence database *SD* is an ordered list of itemsets which is denoted as $S =< IS_1, IS_2, ..., IS_k >$. Sequence database, SD is provided as input to the pattern extraction algorithm.

In the framework, every component is associated with a *utility*.

Definition 2.3.1. *The utility of an item i in an itemset IS of a sequence S can depend on S or IS, in addition to some external factors independent of the database. The*

former is commonly represented with internal utility, denoted as $iu(i, IS, S)$*, and the latter external utility, denoted as* $eu(i)$ *[59].*

Generally, external utilities are specified within an external utility table, and internal utilities are provided within the database sequences. For the exemplification of the ideas, a toy sequence database together with the corresponding external utility table is provided in Tables 2.2 and 2.3 respectively, that will be used throughout this thesis. In Table 2.2, each item is shown together with its *iu* value. For the sake of readability, itemsets are separated by ";" and itemsets longer than of length *2* are shown in square-brackets.

The *size of an itemset IS* of a sequence *S*, denoted as $size(IS, S)$, represents the number of the items in *IS*. For the example provided, $size(IS_2, S_2) = 2$ and $size(IS_2, S_4) = 1$. The *size of a sequence S*, denoted as $size(S)$, represents the number of the itemsets in *S*. For our running example, $size(S_4) = 5$ and $size(S_6) = 2$. The *length of a sequence S*, denoted as $len(S)$, represents the number of items in the sequence. For example, $len(S_1) = 5$, $len(S_2) = 6$ and $len(S_6) = 3$.

Definition 2.3.2. *Given itemsets* $IS_a = \{i_{a_1}, i_{a_2}, ..., i_{a_n}\}$ *of a sequence* S_{c_1} *and* $IS_b = \{i_{b_1}, i_{b_2}, ..., i_{b_m}\}$ *of a sequence* S_{c_2}*;* IS_a *contains* IS_b *if and only if there exists positive integers k, j, z such that; for* $n \geq m$*,* $k \geq 1$*,* $j > k$*,* $z > j$*,* $j, z \leq n$*,* $i_{b_1} = i_{a_k}$*,* $iu(i_{b_1}, IS_b, S_{c_2}) = iu(i_{a_k}, IS_a, S_{c_1})$*,* $i_{b_2} = i_{a_j}$*,* $iu(i_{b_2}, IS_b, S_{c_2}) = iu(i_{a_j}, IS_a, S_{c_1})$*,.....,*$i_{b_m} = i_{a_z}$*,* $iu(i_{b_m}, IS_b, S_{c_2}) = iu(i_{a_z}, IS_a, S_{c_1})$*.*

For instance, itemset $[(a, 4)(a, 1)(c, 2)]$ contains itemsets $(a, 4)$, and $[(a, 4)(c, 2)]$, but it does not contain itemsets $[(a, 4)(c, 4)]$ and $[(e, 1)]$.

Definition 2.3.3. *Given two sequences* $S_a =< IS_{a_1}, IS_{a_2}, ..., IS_{a_n} >$*,* $S_b =< IS_{b_1}, IS_{b_2}, ..., IS_{b_m} >$*;* S_a *contains* S_b *if and only if there exists positive integers i, j, z such that for* $n \geq m$*,* $i \geq 1$*,* $j > i$*,* $z > j$*,* $j, z \leq n$*,* IS_{a_i} *contains* IS_{b_1}*,* IS_{a_j} *contains* IS_{b_2}*,.....,* IS_{a_z} *contains* IS_{b_m}*.*

For instance, sequence $< (h, 5); [(c, 2)(d, 1)] >$ contains $< (h, 5); (d, 1) >$ but does not contain the sequence $< (h, 5); (d, 1); (a, 4) >$.

2.3.2 Utility Calculations

All the components of the high utility sequence mining problem are associated with a utility. In this section, the way the utility calculations are performed in utility based sequential pattern mining framework are defined and formalized.

In a generic framework, the utility of an item and itemset is defined as a function of internal and external utilities of the items [61]. However, in the literature, generally the utility of an item and utility of an itemset are used as given below. In the processing, once the calculated utilities of the items are stored, external utility table is not looked up anymore.

Definition 2.3.4. *The utility of an item is defined as the product of internal and external utilities,* $u(i, IS, S) = iu(i, IS, S) \times eu(i)$.

Definition 2.3.5. *The utility of an itemset is defined as the sum of the utilities of all the items in it.*

Definition 2.3.6. *The sequence utility is calculated by summing the utilities of the itemsets in it.*

Definition 2.3.7. *Sequence database utility is defined as the sum of the utilities of all the sequences it contains.*

Considering the example database, note that

S_2 is $< (b, 2); [(c, 1)(d, 3)]; [(b, 1)(d, 2)(e, 4)] >$.
IS_2 of S_2 is $[(c, 1)(d, 3)]$.
S_5 is $< [(a, 1)(c, 2)]; (b, 2); (d, 1); (b, 1) >$.
IS_1 of S_5 is $[(a, 1)(c, 2)]$.
External utilities of items *a*, *b*, *c*, *d*, *e* are *2*, *2*, *4*, *1* and *3*, respectively.

Then, $u(d, IS_2, S_2) = 3$, $u(IS_2, S_2) = 7$, $u(IS_1, S_5) = 10$, $u(S_2) = 27$, $u(S_5) = 17$ and $u(SD) = 156$.

The minimum utility threshold, $minutil$, refers to a specific percentage of the sequence database utility. In other words, the minimum utility threshold is given by

the percentage of the sequence database utility calculated using ϵ, which specifies the fraction of the total utility of the sequence database. Therefore, $minutil = \epsilon \times u(SD)$.

Considering the example database, since $u(SD) = 156$, if ϵ is *0.1*, then the minimum utility threshold is, $minutil = 0.1 \times 156 = 15.6$.

Pattern utilities should be defined with respect to the database sequences. A pattern can be contained in a sequence more than once due to different ordered lists of itemsets, which should be counted in the definition of pattern utility. This results in a pattern to have a set of utility values with respect to a sequence, which adds further complexity to the high utility sequential pattern mining problem. We formalized this property in the definitions below, before formulating the pattern utility.

Definition 2.3.8. *Let $P =< p_1, ..., p_k >$ be a pattern of k itemsets and S be a sequence. An ordering of P in S is defined as an ordered list of k itemsets, $< IS_a, IS_b, ..., IS_z >$ of S such that, $a < b < ... < z$, the first itemset of P is contained in IS_a, the second itemset of P is contained in IS_b, ..., and the k^{th} itemset of P is contained in IS_z.*

For instance, consider the pattern $< bd >$ and the example database. The pattern is contained twice in S_2 with two orderings $O_1 =< IS_1, IS_2 >$, and $O_2 =< IS_1, IS_3 >$.

Orderings of a pattern in a sequence are ordered among themselves, which is formalized below.

Definition 2.3.9. *For any two orderings, $O_i =< IS_{i_1}, ..., IS_{i_k} >$ and $O_j =< IS_{j_1}, ..., IS_{j_k} >$ of a k-sized pattern P in S, O_i precedes O_j in S, denoted as $O_i < O_j$, if IS_{i_k} and IS_{j_k} are the same itemsets or IS_{i_k} precedes IS_{j_k} in S.*

For instance, for the pattern $< bd >$ and the orderings O_1 and O_2, $O_1 < O_2$ since IS_2 precedes IS_3.

Using these definitions, we formalize the calculation of the pattern utility as given below.

Definition 2.3.10. *The utility of a pattern* P *in a sequence* S *with respect to an ordering* O *is denoted as u(P,S,O), and defined as the sum of the utilities of the items in the*

itemsets included in the corresponding ordering.

As an example, note that S_2 is $< (b,2);[(c,1)(d,3)];[(b,1)(d,2)(e,4)] >$ and external utilities of b and d are 2 and 1, respectively. Then, for the pattern $< bd >$ and the two orderings O_1 and O_2 in S_2, $u(< bd >, S_2, O_1) = 7$ and $u(< bd >, S_2, O_2) = 6$.

Definition 2.3.11. *The utility of a pattern* P *that has m orderings in a sequence S is denoted as* u(P,S)*, and defined as in Equation 2.1.*

$$\max\{u(P, S, O_k) : k = 1, \ldots, m\} \tag{2.1}$$

For $< bd >$, $u(< bd >, S_2)$ is therefore 7.

Definition 2.3.12. *The utility of a pattern* P *with respect to a sequence database* SD *of n sequences is defined as in Equation 2.2.*

$$u(P) = \sum_{d=1}^{n} u(P, S_d) \tag{2.2}$$

For instance, considering the example database, $< bd >$ is contained in S_2, S_5 and S_6, where $u(< bd >, S_2) = 7$ and $u(< bd >, S_5) = 5$ and $u(< bd >, S_6) = 10$. Therefore $u(< bd >) = 22$.

2.3.3 Formalization the Problem of High Utility Sequential Pattern Mining

Considering all the concepts defined above, the problem of high utility sequential pattern mining is formalized as follows: *Given a utility sequence database SD and a minimum utility threshold* $minutil$*, high utility sequential pattern mining aims to extract all high utility sequences in SD that has utility greater than or equal to* $minutil$.

Considering the example database, if $minutil$ is 15.6, $< bd >$ is a high utility pattern since $u(< bd >) = 22$, which is greater than 15.6. Considering pattern $< (cd) >$, which is contained in only S_2 with a single ordering, $u(< (cd) >) = 7$. Therefore, $< (cd) >$ is not considered as a high utility sequential pattern.

2.3.4 Research Challenges

Mining high utility sequential patterns is a more complex task in comparison to frequent pattern mining and high utility itemset mining due to two main challenges. First of all, compared to frequent pattern mining, *Downward Closure Property* does not hold, when utilities is of concern [59]. In addition, sequencing between itemsets results in a huge combinatorial search space, which leads to high computational complexity. In order to tackle with the first challenge, existing studies [61, 36] incorporate the concept of *TWU (Transaction Weighted Utilization)* [36]. TWU of a candidate pattern is defined as the sum of the utilities of all the transactions containing that pattern. Here, the utility of a transaction is calculated as the sum of the utilities of all the items in it. A pattern is considered as a candidate or potential high utility pattern, if its TWU is not less than a minimum utility threshold. However, the main problem of TWU-based pruning is that, it results in a large number of candidates, since the utilities of all the items, even known to be not included in the parent pattern, are counted in the calculation of the upper bound. This results in calculating overestimated utilities, which in turn results in a degraded mining performance.

The success of any high utility sequential pattern mining solution depends on its power to restrict the number of the candidates and simplify the computation for calculating the utility [36]. A *tighter upper bound* on the utility of candidate patterns may result in pruning the search space better. In addition, the phase of the solutions at which the pruning mechanism is used, effects the performance of the algorithms. Pruning the search space can be performed either prior to or after candidate generation, which are referred to as *Pruning Before Candidate Generation (PBCG)* and *Pruning After Candidate Generation (PACG)*, respectively. In comparison to PACG, PBCG saves more space and time, since candidates are eliminated before they are generated and tested. Moreover, most of the computational complexity of pattern extraction lies in the calculation of the utilities, and to lower this complexity, efficient data structures should be utilized.

Although utility-based sequential pattern mining provides a solution for the limitations of frequent sequential pattern mining, it does not cover all real-life scenarios. In other words, existing utility-based pattern mining techniques calculate pattern utilities

based on the utility of the individual items. However, a pattern's value cannot always be calculated from the distinct item utilities. Instead, they should be calculated for each extracted pattern on its own. In addition, common to both frequency-based and utility-based techniques, one important challenge is that, there exists a huge number of possible patterns and a complete algorithm should find the set of all patterns, when possible, satisfying the minimum support or the minimum utility threshold. However, it is not possible to extract patterns under high threshold values, and for low utilities, the existing algorithms fail to respond especially for huge and sparse datasets.

2.4 Web Usage Mining

The second solution proposed in the thesis work aims to extract sequential patterns by utilizing user-defined scoring function. Although the proposed solution can be adapted to work with any sequential pattern mining data, for the sake of experimenting with the developed framework, in the proposed version, web usage mining domain is used and therefore, the solution is adapted to work with web logs.

Web log data is the file or several files that are automatically created and maintained by a server. Web logs keep the information of a history of *page requests*. Each request comes from a *user* and requests from the same user are divided into *sessions* based on the IP and duration of the requests.

A Web server log file contains requests made to the web server, recorded in a chronological order. Therefore, it is a sequential data, in which the items are page requests that are ordered by time they are sent to the web server. The most popular log file formats are the *Common Log Format*[1] and an extended version of the Common Log Format, *Combined Log Format*. A line in the extended version of the Common Log Format contains the following information:

- client's host name or IP address,
- user login if exists,
- date and time of the request,

[1] http://www.w3.org/Daemon/User/Config/Logging.html#common-logfile-format

- operation type like GET, POST, HEAD, etc.,
- requested resource name,
- status,
- requested page size,
- user agent which is a string identifying the browser and the operating system used,
- referrer of the request which is the URL of the Web page containing the link that the user followed to get to the current page.

Web Usage Mining process includes three main steps: *preprocessing, pattern discovery* and *pattern analysis* [17]. The preprocessing task is very important for effective pattern discovery from web log data, since, the web usage data generally includes a lot of noise and missing data. The preprocessing task involves cleaning and structuring the data to prepare it for the pattern discovery task. In order to perform this, all irrelevant and noisy data should be eliminated from the log files. This usually includes removing the requests for images and multimedia files as well as web robots' sessions. When requesting a web page containing additional web resources like images or script files, several implicit requests will be generated by the web browser. If these requests are not removed before the data mining step, uninteresting patterns may be found, making the pattern analysis step more complex. For instance, considering web robots, they usually have a predefined and programmed behavior such that, the analysts are not interested in mining these requests. Therefore, such requests should also be cleared in addition to erroneous accesses and implicit requests.

After the cleaning process is completed, the resulting data should be structured for pattern discovery. In order to achieve this, requests from the raw log file are grouped by user, user session, page view, visit and episode depending on the needs of the developed application. Grouping requests by user, also called *user identification*, depends on the web site policies. For instance, if the web site whose log files are examined, requests registration, the user identification task is straightforward and guaranteed to be correct. However, for all other methods, such as identifying users by using the IP address, the accuracy of the user identification is not guaranteed.

The user session contains all the requests of a user, made from the same IP address and having the same user agent. The page view identification is important, since it results in selecting the explicit requests from the web access log file. The page view is the response of the web browser to the user click. Usually, the web browser generates several implicit requests for each explicit web request such as images, multimedia files or other HTML files. At the end of the page view identification process, only one request per page view has to be kept, which is the explicit request made by the user. Visits and episodes identification depend on the purpose of the analysis. For some analysis, grouping the requests in user sessions and keeping one request per page view is enough and there is no need to further split the user session. However, one user session may span over several months, and in such a case, the requests will not be related. Therefore, it is better to group the requested pages in episodes according to their content and analyze them separately.

When the raw web server log files have been preprocessed, data mining techniques can be applied on the dataset to discover new patterns. There have been many studies in order to extract patterns from web log data which are briefly mentioned in Chapter 3. These techniques include, but are not limited to association rule mining, sequential pattern mining and clustering. The scope of this thesis is sequential pattern mining and the developed solution aims to discover patterns from web server log files satisfying a user-defined evaluation function.

After pattern discovery, pattern analysis step aims at helping the user to analyze the results by providing a visualization and request tool, which is beyond the scope of this thesis.

From the sequential pattern mining point of view, web usage data is also a sequence data such that, requests made by the user in a session are ordered by the time they are made. In addition, distinct user sessions are also ordered among each other. In order to map the web server log files for utility-based and user-scoring based pattern extraction process, two different mappings can be performed. First, each session can be considered as a distinct sequence and all the itemsets in a sequence contain a single item. Second, all the sessions of a user can be treated as a transaction and each of the distinct sessions of a user can be treated as an itemset. Here, in this second mapping,

the sequencing information between web pages in the same session is lost.

The first proposed solution is a technique developed for utility-based pattern extraction where in the evaluation phase, datasets from different domains have been utilized. The second solution, however, is adapted to web domain and evaluated using web server log files. Therefore, the details about how the data cleaning and data structuring steps for the log files is performed in the proposed solution is described in Chapter 5.

CHAPTER 3

RELATED WORK

This section provides an overview of recent studies in an attempt to outline the approaches and main findings in sequential pattern mining, utility based mining and web usage mining research so as to both understand and analyze the literature and better outline the main contributions of the thesis work by comparing it with the state of art techniques from related domains.

3.1 Sequential Pattern Mining

Sequential pattern mining is an important data mining problem with broad application areas including telecommunication industry, retail industry to analyze the sales data, medical applications and stock market analysis. The sequential pattern mining solutions aims to find the complete set of frequent subsequences given a sequence database and support threshold. The challenges related to sequential pattern mining can be stated as follows. First, a huge number of possible sequential patterns are hidden in databases and a mining algorithm should find the complete set of patterns, when possible, satisfying the minimum support threshold. Second, the patterns can be formed using single items as well as itemsets, therefore, the possible number of sequences are large in comparison to itemset mining. Third, the sequential pattern mining solutions should be highly efficient, scalable, involving only a small number of database scans and they should be able to incorporate various kinds of user-specific constraints if needed [28].

There has been many different approaches [54, 2, 40, 64, 65, 12] proposed for se-

quential pattern mining problem. The solutions differ in the way by which candidate sequences are created and stored in memory, the support value is calculated and how the candidate sequences are tested by using this support value. Based on these criteria, sequential pattern mining solutions can be classified into two groups, namely, *Apriori Based* Approaches and *Pattern Growth Based* Approaches [38].

3.1.1 Apriori-Based Approaches

The Apriori-based approaches are extensions of the Apriori-based frequent pattern mining algorithm [1] to sequential pattern analysis. The Apriori and AprioriAll algorithms form the basis for this group of sequential pattern mining algorithms that depends largely on the *Apriori Property* to generate the candidate sequences. The Apriori Property of sequences states that, *if a sequence S is not frequent, then none of the super-sequences of S can be frequent*. This property is also referred to as the *Downward Closure Property*. These algorithms identify the frequent itemsets in the database and extend them to a larger itemset as those itemsets appear frequently in the database.

From the sequential pattern mining point of view, a sequence database can be represented in two data formats: a *horizontal data format* and a *vertical data format*. The former uses the natural representation of the dataset as a sequence of objects, whereas the latter uses the vertical representation of the sequence database in the form of objects such that for each object it keeps the sequence id it exists in, together with its order information within the sequence. Based on the way the data is represented, the Apriori-based algorithms are divided into two sub groups; *horizontal data format based sequential pattern mining algorithms* and *vertical data format based sequential pattern mining algorithm*s.

GSP (Generalized Sequential Pattern) is a horizontal data format based sequential pattern mining algorithm developed by the authors of [54]. It is an extension of frequent itemset mining algorithm, Apriori [1]. The algorithm adopts a multiple-pass, candidate generation and test approach based on the DCP of a sequential pattern. GSP is outlined as follows. First, the database is scanned in order to find all the frequent items, which form the set of frequent sequences of single items. Each subsequent

scan starts with a set of sequential patterns, which is the set of patterns found in the previous pass. This set is used to generate new potential patterns, called *candidate sequences*. Each candidate sequence contains one more item than the set of sequential patterns from the previous pass, where each element in the pattern may contain one or multiple items. All the candidate patterns in a pass have same length and the scan of the database in one pass finds the support for each of the candidate patterns. The candidates whose support is not less than the minimum support threshold form the set of the newly found sequential patterns for the next pass. The algorithm terminates when no new sequential pattern is found, or no candidate pattern can be generated.

The main drawbacks of the GSP algorithm include the generation of a huge set of candidate patterns, and performing multiple scans of the database. Therefore, the algorithm is inefficient for mining long sequential patterns, as it needs to generate a large number of small candidates in terms of pattern length.

In [65], a vertical data format based sequential pattern mining algorithm, *SPADE (Sequential PAttern Discovery using Equivalent classes)*, is proposed that takes a different approach than Apriori and GSP for the discovery of sequential patterns. Instead of assuming a horizontal database, SPADE assumes a vertical database, where each item is associated with a sequence id and an ordering. The support of each of the *k-sequences* is determined by the temporal join of *k-1* sequences that share a common suffix. SPADE reduces the number of database scans, since the information required to construct longer sequences are localized to the related items, and subsequences are represented by their associated sequence and ordering identifiers. However, the basic search methodology of SPADE is similar to GSP, which is based on breadth-first search and Apriori-based pruning. Therefore, similar to GSP, SPADE has to generate a large set of candidates in breadth-first manner in order to grow longer subsequences.

SPAM (Sequential Pattern Mining) [12] is another solution to sequential pattern mining problem which uses a vertical bitmap data structure representation of database that is similar to the data representation of SPADE. One important property of SPAM is that, it is the first sequential pattern mining method that utilizes a depth-first search approach to explore the search space. SPAM combines depth-first search strategy with an effective pruning technique, which reduces the number of candidates effi-

ciently. From this perspective, the algorithm is particularly suitable for long sequential patterns. However, SPAM requires the whole database to be stored in the memory, which is the main drawback of the algorithm.

3.1.2 Pattern-Growth Based Approaches

Pattern-growth is a method of frequent pattern mining that does not require candidate generation [29]. The technique originated in the *FP-growth* algorithm [30] for transaction databases and the general idea of pattern-growth based sequential pattern mining is as follows: First, frequent single items are found and this information is compressed into a frequent-pattern tree, or *FP-tree*. The FP-tree is then used to generate a set of projected databases, such that, each database is associated with one frequent item. These databases are mined separately and prefix patterns are built by concatenating them with suffix patterns to construct frequent patterns, avoiding candidate generation.

FreeSpan (Frequent Pattern Projected Sequential Pattern Mining) [30] is an algorithm introduced for the reduction of candidate generation and testing through level-wise sequential pattern mining. The algorithm uses the frequent items to recursively project sequence database into a set of smaller projected databases. Thus, the execution time of the algorithm is improved, because each projected database is smaller and easier to treat.

In [47], authors propose a pattern-growth based technique for sequential pattern mining, called *PrefixSpan*, that deviates from the standard ways of generating candidates and testing, as it is the case for GSP and SPADE. The proposed algorithm is projection-based, such that, the sequence database is iteratively projected into smaller projected databases. PrefixSpan grows patterns sequentially in the projected databases by investigating only locally frequent segments. Specifically, in a database scan, the frequent patterns above a certain support are recorded. For each frequent pattern, its ending location is used as the beginning location of the new scan. Frequent patterns are then discovered and appended to the frequent patterns discovered in the previous scan.

One of the computational costs of PrefixSpan is the creation of the projected databases. In [47], authors propose *pseudoprojection* as a way to reduce the number and size of the projected databases, where the index of the sequence and the starting position of the projected suffix are recorded, instead of performing a physical projection.

A variation of PrefixScan, namely, *CloSpan*, is proposed in [58] which detects closed subsequences. A closed sequential pattern is one for which there is no superset sequence with the same support than contains the pattern.

WAP-mine is a pattern-growth and tree structure mining technique with its WAP-tree structure [52]. The algorithm scans the sequence database twice to build the WAP-tree from frequent sequences along with their support. A header table is maintained to point at the first occurrence for each item in a frequent itemset, which is later tracked in a threaded way to mine the tree for frequent sequences. The algorithm is reported to have better scalability than GSP and to outperform it by a margin. Although WAP-mine scans the database only twice and can avoid the problem of generating explosive candidates as it is the case for Apriori-based methods, it suffers from the memory consumption problem, as it recursively reconstructs numerous intermediate WAP-trees during mining.

Although numerous scalable methods have been developed for mining frequent sequences, these solutions often generate a huge number of frequent patterns. However, the target users would like to see or use only the interesting patterns whose value is based on some other factors in addition to frequency. Many recent studies have contributed to mining interesting patterns or rules, including *constraint-based mining*, and *utility-based mining* which will be covered in the following subsections.

3.2 Constraint-Based Sequential Pattern Mining

The frequency-based mining solutions generally return a huge number of patterns, many of which could be uninteresting to users. In addition, these algorithms often takes substantial computational time and space for mining the complete set of sequential patterns in a large sequence database. To prevent these problems, users can use constraint-based sequential pattern mining for focused mining of desired patterns.

Constraints could be antimonotone, monotone, succinct, convertible or inconvertible [45]. Anti-monotonicity means that if an itemset does not satisfy the rule constraint, then none of its supersets satisfy. Monotonicity, on the other hand, refers to the following; if an itemset satisfies the rule constraint, then all of its supersets satisfy that rule constraint. Succinctness means that, all and only those patterns guaranteed to satisfy the rule can be enumerated. Convertible constraints are those, which are not antimonotonic, monotonic, or succinct, but can be made anti-monotonic or monotonic by changing order of elements in the set. Inconvertible constraints, on the other hand, are the ones which are not convertible.

In the context of constraint-based sequential pattern mining, the scope of the Apriori-based sequential pattern mining is generalized to include time constraints, sliding time windows, and user-defined taxonomy [54]. Since episodes are essentially constraints on events in the form of acyclic graphs, mining frequent episodes in a sequence of events [39] can also be viewed as a constrained mining problem. The classical framework on frequent and sequential pattern mining is based on the anti-monotonic Apriori property of frequent patterns. A breadth-first, level-by-level search can be conducted to find the complete set of patterns. Performance of conventional constraint-based sequential pattern mining algorithms dramatically degrades in the case of mining long sequential patterns in dense databases or when using low minimum supports. Weight constraints are used in order to reduce the number of unimportant patterns in [63], where during the mining process, not only the supports but also the weights of patterns are considered.

User-defined tough aggregate constraints are incorporated in [16], so that, the discovered knowledge better meets user needs. A novel algorithm, namely, *PTAC (sequential frequent Patterns mining with Tough Aggregate Constraints)* is proposed in order to reduce the cost of using tough aggregate constraints by incorporating effective strategies.

In [41], *GTC (Graph for Time Constraints)* technique is presented for mining time constraint-based patterns in very large databases. The solution is based on the idea that, handling time constraints in the earlier stage of the data mining process can be highly beneficial. One of the most significant new features of this approach is that,

handling of time constraints can be easily taken into account in traditional level-wise approaches, since it is carried out prior to and separately from the counting step of a data sequence.

Approximate structural patterns from a genetic sequences database is examined in [57]. The described technique allows the users to specify the desired form of patterns as sequences of consecutive symbols separated by variable length don't cares, in addition to a lower bound on the length of the discovered patterns, and an upper bound on the edit distance allowed between a mined pattern and the data sequence that contains it. A random sample of the input sequences to build a main memory data structure is used, which is called the *generalized suffix tree*. This tree is then used to obtain an initial set of candidate pattern segments and screen out candidates that are unlikely to be frequent based on their occurrence counts in the sample.

In [25], regular expressions are used as constraints for sequential pattern mining and the authors developed a family of *SPIRIT (Sequential Pattern Mining with Regular Expression Constraints)* algorithms. Members in the family achieve various degrees of constraint enforcement. The algorithms use relaxed constraints with properties like anti-monotonicity to filter out some unpromising candidates in their early stage.

The main limitation of the constraint-based solutions is that, the users may be interested in high-valued patterns and the value definition cannot always be handled by filtering patterns based on specified constraints, which is solved by the utility-based mining techniques.

3.3 Utility-Based Mining

The limitations of frequency-based and constraint-based pattern mining solutions motivated researchers to conceive a utility-based mining approach, which allows a user to express his or her perspectives concerning the usefulness of patterns as utility values, and then find patterns with utility values not less than a specified minimum utility threshold. In high utility pattern mining, the meaning of utility refers to the interestingness, importance, or profitability of the patterns. Studies in this area generally focus on *mining high utility itemsets from transactional databases* [59, 36, 6, 55],

and *high utility sequential pattern mining from sequence databases* [61, 60, 50, 3].

3.3.1 Utility-Based Itemset Mining

Mining high utility itemsets from databases was first introduced in [59]. Utility of items in a transaction database generally considered to include two main aspects: first, the importance of distinct items, which is called as *external utility*, and second, the importance of the items in transactions, which is referred to as *internal utility*. Utility of an item is then defined as the product of its external and internal utilities [59]. The itemset utility is defined as the sum of the utilities of all the items in it. An itemset is considered as a high utility itemset if its utility is no less than a user-specified minimum utility threshold.

Several algorithms are proposed for high utility itemset mining including *UMining* [60], *IHUP* [6], and *UP-Growth* [55]. UMining is a well known efficient algorithm for mining high utility itemsets from large transaction databases, however, it cannot extract the complete set of patterns.

In order to extract high utility patterns more efficiently, *Transaction-Weighted Downward Closure (TWDC)* property was introduced in [36]. TWDC uses *Transaction Weighted Utilization (TWU)* of the patterns in order to prune the search space. Most of the following studies also adapt TWU in their solutions. A tree-based algorithm, *IHUP* [6] is proposed by Ahmed et al. to avoid scanning database too many times. IHUP uses IHUP-Tree to maintain information about itemsets and their utilities. UP-Growth, on the other hand, also uses a tree structure, to mine high utility patterns; and it is reported as a more efficient solution than IHUP, since it further reduces the number of promising patterns, which cannot be pruned in IHUP. UP-Growth and UP-Growth+, described in [56], outperforms other algorithms in terms of execution time, especially when databases contain lots of long transactions or when low minimum utility threshold values are set.

3.3.2 Utility-Based Sequence Mining

The main shortcoming of high utility itemset mining solutions is that, although they perform well in many applications, they cannot answer the needs in domains where the ordering between itemsets is important. The addition of ordering information in high utility sequential pattern mining makes the pattern mining problem fundamentally different and much more challenging than mining high utility itemsets. The integration of utility and sequential pattern mining has been attracting attention recently and not many solutions exist in this area [61, 50, 3, 4, 62]. In [50], *UMSP* algorithm is proposed for mining high utility mobile sequential patterns in mobile environments. In UMSP, the utility of a mobile sequential pattern is defined as a single value such that, it relates each itemset in a sequence with a location identifier. UMSP searches for patterns using an efficient structure, called *MTS-Tree*. However, the main limitation of this solution is that, it can handle only the sequences of single items and single utility value per item is permitted.

In [3], an algorithm for extracting high utility sequential patterns from web log sequences is proposed. As opposed to UMSP, the utility of a pattern can have multiple values, and in such a case, the proposed technique selects utility with the maximal value. The utilities are represented with two tree structures, *UWAS-tree* and *IUWAS-tree*. However, the study does not support the sequencing of the itemsets. In [4], the *Sequence-Weighted Utility (SWU)* measure is defined which is used to satisfy the Downward Closure Property for mining high utility sequential patterns. In *SWU*, the idea behind is very similar to that of *TWU* such that, *SWU* is also defined as the sum of the utilities of all the transactions containing the pattern. In other words, *SWU* is the sequence-weighted counterpart of *TWU*. In addition, in [4], authors proposed *UtilityLevel (UL)* and *UtilitySpan (US)* algorithms. UL algorithm follows a candidate generation approach, and the US algorithm performs high utility sequential pattern mining with a pattern-growth approach. However, the problem definition is rather specific and no generic framework is proposed.

A projection-based approach for mining high utility sequential patterns with the maximum utility measure and a sequence-utility upper-bound (SUUB) model are proposed in [34]. The main concept of the proposed method is extended from the PrefixSpan

algorithm [46]. In [62], a different problem is addressed in the area of high utility sequential pattern extraction, where the authors aim to identify the *top-k* high utility sequential patterns without minimum utility. The top-k pattern extraction is especially useful when it is difficult to set a minimum utility threshold in some domains and when the pattern users are sure about the number of extracted patterns that they will investigate. However, for the datasets where the patterns are stacked around a utility threshold, it will not be appropriate to choose top k of them when there exists many patterns with very similar utilities.

Considering the pruning mechanisms, existing studies [61, 36] for high utility sequential pattern mining use *TWU*-based upper bounds during PBCG. In [61], a general framework for mining sequential patterns from utility-based sequence databases is described. In order to extract patterns, they propose *USpan* algorithm which uses *SWU* as defined in [4] and *Sequence Weighted Downward Closure (SDCP)*. *SDCP* utilizes *SWU* to select candidate items during pattern generation as follows. Based on *SWU* of a pattern, an item is called *promising* if adding it to a candidate pattern results in a new pattern whose *SWU* is greater than or equal to the minimum utility threshold. In such a case, this candidate pattern is generated and it appears in the resulting pattern tree. In USpan, PBCG is based on *SDCP* and the algorithm performs depth pruning which is a PACG mechanism.

Although SDCP correctly eliminates candidate items, a tighter upper bound may eliminate more promising items, and performing such a pruning prior to candidate generation may reduce the required time and space, which is the main motivation of our study. To achieve this, we propose CRoM, and use it in the pruning of the candidates. We show that, CRoM correctly eliminates promising items. In comparison to the existing solutions mentioned for utility-based sequential pattern mining, our proposed pruning technique is used prior to candidate generation and defines a tighter upper bound than TWU-based pruning before generation of the candidate patterns. Moreover, we propose data structures to perform both the pruning and the candidate generation processes efficiently. We realized our solution with a new algorithm, *HuspExt*, which is based on a generic framework that provides solution for any kind of data from any domain. The utilities of items, itemsets, and sequences are considered as functions that can be defined and specified by the domain experts according to the

needs of the application.

3.4 Web Usage Mining

There is a huge growth of information sources on the Internet and together with this growth, the user base also grows. In such a situation, it is very important and necessary to facilitate users' navigation in the web. Web logs are valuable resources to learn about navigation behaviors. Analysis of web log files to extract useful patterns is studied under *web usage mining* [35], which uses various techniques including *clustering, association rule mining, classification and sequential pattern mining* [43]. The extracted patterns are web access patterns that are pursued by the web users frequently. Web access patterns are generally used for recommendation, next page prediction and web site improvement purposes [26], [10], [7], [8].

Web usage mining has been studied by various researchers and different techniques [42], [27], [33] have been applied to process click-stream data. In 1996, O. Etzioni [21] explored the question of whether effective web mining is feasible in practice. He believed that the web is too unstructured for web mining to succeed. In the recent years, however, there has been an increase in the number of studies on web usage mining to examine the users' web navigation behavior for improving the quality of the web services offered to the web users [51, 66, 53, 32, 68]. The main motivation of these studies is to get a better understanding of the reactions and motivations during web navigation and to help the users in their interaction with the web site.

Learning users' navigation behaviors for web page recommendation has been studied by the researchers and various attempts have been exploited to achieve web page access prediction [42, 27, 33, 19]. Web data clustering has also been studied by different researchers and solutions in this area are generally grouped into (i) *sessions-based* [13], [24] and (ii) *link-based* [20, 23]. The former aims to group users' navigation sessions having similar characteristics. The latter, however, treats the web site as a directed graph and the goal is to cluster the web pages with similar content.

Sequential pattern mining solutions are also applied to web access pattern discovery [49, 14]. However, these solutions cannot respond to cases where different web pages

have different significance values. To handle this problem, utility-based web access pattern mining techniques have been proposed [67, 5, 3]. In [67], authors adopted the definitions of utility from the high utility pattern mining model. The browsing time of a web user constitutes the internal utility of a web page. By doing so, more interesting web traversal patterns can be discovered in comparison to the classical frequency based solutions.

Existing utility-based web access pattern mining techniques calculate pattern utilities based on the utility of the web pages. However, a pattern's value cannot always be calculated from the distinct web page utilities. For instance, user may be interested in patterns that exist in at least for a number of distinct users' sessions in the dataset. Similarly, the value of a pattern may depend on the average income of the users that traverse that pattern. Such values cannot be calculated from the web page utilities and should be recalculated for each extracted pattern on its own.

Motivated by these real-life scenarios, in the thesis work, we present a new approach to the problem of extracting web access patterns. Different from the existing techniques in literature, the proposed solution combines clustering with a search-based pattern extraction algorithm which uses an evaluation function that can be defined according to the preferences or needs of the users. Different from the existing techniques in this area, pattern extraction process in the proposed solution is not based on clustering. Instead, clustering is utilized so as to improve the efficiency and the accuracy of the pattern extraction algorithms by identifying different behavioral pattern groups.

CHAPTER 4

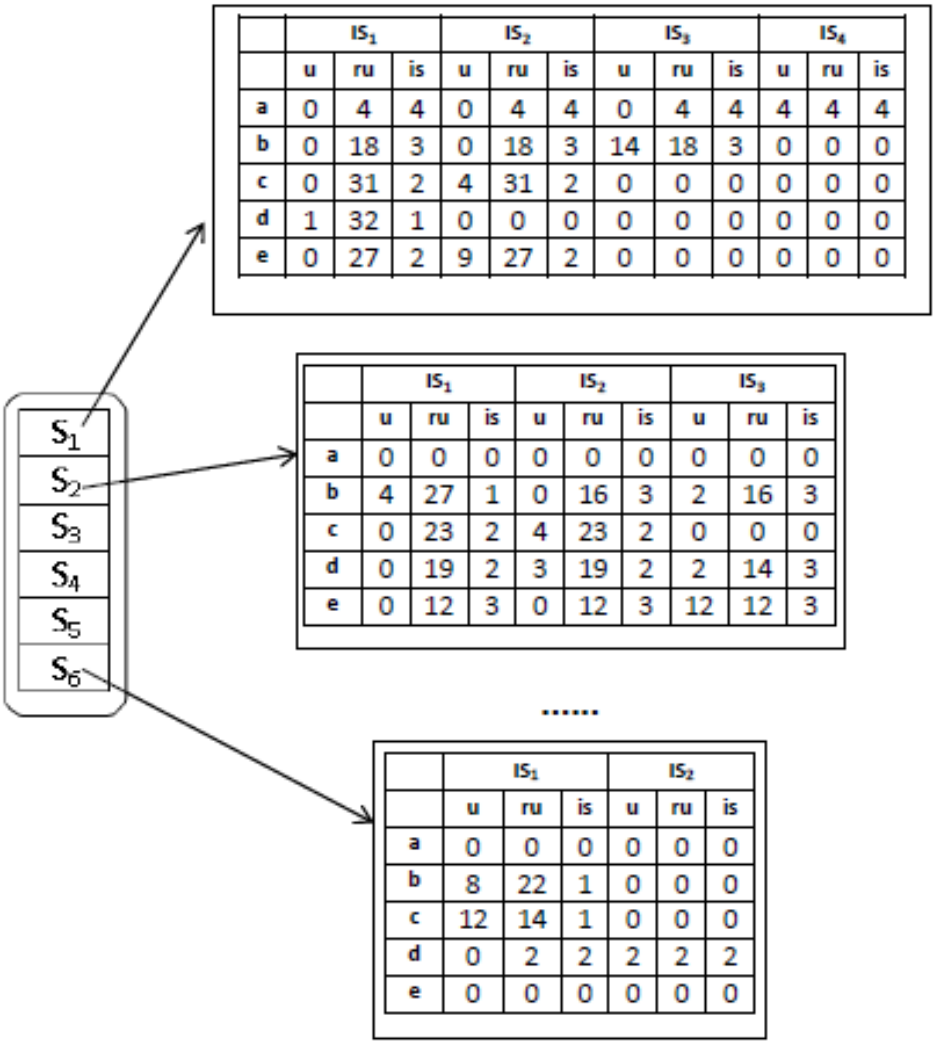
IMPROVING EFFICIENCY OF HIGH UTILITY SEQUENTIAL PATTERN EXTRACTION

In this chapter, proposed solution for high utility sequential pattern extraction is described by giving details about the structure of the database sequences and patterns, construction of the pattern tree, pruning strategy that uses CRoM based upper bound, and the HuspExt algorithm [11].

4.1 Structure of the Database Sequences

In the proposed solution, each database sequence is stored as a *data matrix S*. Figure 4.1 presents the data representation in HuspExt for the example database given in Tables 2.2 and 2.3.

The columns of the matrix are the itemsets and the rows are the items in the database. Each cell $S(i, j)$ of the data matrix for a sequence S keeps the utility information about item *i* in the IS_j of the sequence S. The utility information is kept as a triple *(u,ru,is)*, which is used by HuspExt in the pruning process. *u* is the utility of the item, that is, *u(i,* IS_j*, S)*. $S^u(i, j)$ equals to 0 if item *i* does not exist in IS_j of S. $S^{ru}(i, j)$ is the sum of the utilities of the rest of the items in the sequence including the utility of the item itself, if $S^u(i, j)$ is greater than 0. If it is 0, then $S^{ru}(i, j)$ equals to the *ru* value of the item *i* in the first following itemset that it exists. This itemset information is kept in $S^{is}(i, j)$. Therefore, if $S^u(i, j)$ is 0, then $S^{is}(i, j)$ keeps the first itemset that has item *i*. If the item is not contained in any of the following itemsets, then both *ru* and *is* are equal to 0.



	IS1			IS2			IS3			IS4		
	u	ru	is	u	ru	is	u	ru	is	u	ru	is
a	0	4	4	0	4	4	0	4	4	4	4	4
b	0	18	3	0	18	3	14	18	3	0	0	0
c	0	31	2	4	31	2	0	0	0	0	0	0
d	1	32	1	0	0	0	0	0	0	0	0	0
e	0	27	2	9	27	2	0	0	0	0	0	0

	IS1			IS2			IS3		
	u	ru	is	u	ru	is	u	ru	is
a	0	0	0	0	0	0	0	0	0
b	4	27	1	0	16	3	2	16	3
c	0	23	2	4	23	2	0	0	0
d	0	19	2	3	19	2	2	14	3
e	0	12	3	0	12	3	12	12	3

	IS1			IS2		
	u	ru	is	u	ru	is
a	0	0	0	0	0	0
b	8	22	1	0	0	0
c	12	14	1	0	0	0
d	0	2	2	2	2	2
e	0	0	0	0	0	0

Figure 4.1: Structure of the Database for the example in Table 2.2.

In order to clarify the idea, consider S_2 in the example database. Figure 4.1 clearly shows the structure of S_2. Item *b* exists in IS_1; therefore $S_2^u(b,1)$ is 4 and $S_2^{is}(b,1)$ is 1. Moreover, $S_2^{ru}(b,1)$ keeps sum of the utilities of the rest of the items following it, including item *b*, which is 27. Furthermore, $S_2^u(d,1)$ is 0, since item *d* does not exist in IS_1; and $S_2^{ru}(d,1)$ equals to the ru of the first occurrence of the item *d* in the rest of the sequence, that is $S_2^{ru}(d,2)$, which is 19. Since *d* exists in IS_2, $S_2^{is}(d,1)$ is 2. ru, u and is are equal to 0, when an item neither exists in the corresponding itemset nor in the rest of the sequence following this itemset.

The calculations of ru and is are done in a single scan of the database. HuspExt starts calculation from the last itemset of each item, and it keeps for each item the last ru value and the corresponding is to assign it, if u equals to 0. The ru and is values are used in the pruning process.

4.2 Structure of the Patterns

In the proposed solution, every pattern P is represented by a 3-tuple $< ps, u, CSeq >$. *ps* denotes the pattern string and *u* denotes the utility of the pattern in the database. *CSeq* is the information about the sequences that contain the pattern. For each ordering O_k of a pattern P in a sequence S, $CSeq(S,k)$ keeps a pair, *($last\_IS$, u)*. $CSeq_P^{last\_IS}(S,k)$ is the itemset of S that matches the last itemset of P in the ordering O_k, and $CSeq_P^u(S,k)$ is the utility of that ordering.

To illustrate the structure, consider the pattern $< bd >$ in the example database, whose structure is provided in Table 4.1. The pattern exists in S_2 with two orderings; $< IS_1, IS_2 >$, and $< IS_1, IS_3 >$, that has utilities 7 and 6, respectively. Therefore, $u(< bd >, S_2) = 7$. Considering S_5, $< IS_2, IS_3 >$ is the only ordering with utility 5, therefore, $u(< bd >, S_5) = 5$. Finally, for S_6, there exists a single ordering, $< IS_1, IS_2 >$, which has utility 10. Therefore, $u(< bd >) = 7 + 5 + 10 = 22$.

Table 4.1: Pattern Structure for $< bd >$

ps			$< bd >$
u			22
CSeq			
Sequence	Ordering	$last\_IS$	u
S_2	1	2	7
S_2	2	3	6
S_5	1	3	5
S_6	1	2	10

4.3 Construction of the Pattern Tree

We use the lexicographic approach [12] during the construction of our pattern tree such that, nodes in the tree are arranged according to their lexicographic order of the pattern string *ps* similar to the way it is used in [61, 36]. However, different from existing solutions, we included *ordering* information and used *CSeq* structure in order to apply CRoM based pruning in HuspExt to efficiently prune the tree and extract sequential patterns.

Pattern tree keeps the patterns that are built throughout the execution of HuspExt. At each level *k* of the tree, HuspExt generates child nodes to be potentially added to level *k+1* by using *item-based expansion* or *itemset-based expansion*. Each expansion is performed by adding a single item to the parent pattern. The item-based expansion generates a child node whose pattern is created by adding an item into the last itemset of the parent's pattern. For instance, if the parent pattern is $< (bd) >$, then the item-based expansion of the parent-pattern with the item *e* results in the child pattern$<$ $(bde) >$. Therefore, item-based expansion results in a pattern whose size is the same as the parent's, however whose length is one item longer than that of the parent's. The itemset-based expansion, on the other hand, generates a new node whose pattern contains a new itemset consisting of a single item added to the end of its parent's pattern. This expansion increases the length and size of the child pattern with respect to the parent pattern by one. For instance, considering parent pattern $< (bd) >$, itemset-based expansion with item *e* results in $< (bd)e >$.

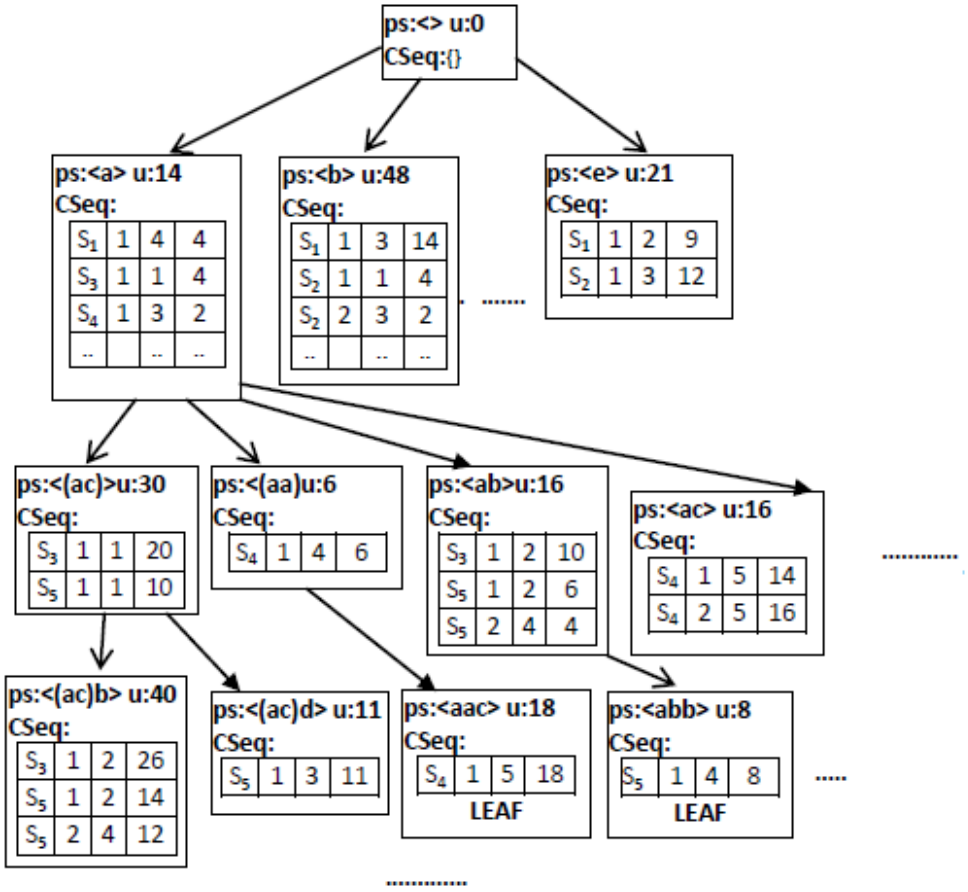


Figure 4.2: Pattern Tree for the example in Table 2.2.

Figure 4.2 shows a sample pattern tree. In this figure, columns of $CSeq$ keeps the sequence, the corresponding ordering, $last\_IS$, and u values respectively. For instance, first row of the $CSeq$ of pattern $< a >$ is $< S_1, 1, 4, 4 >$, which refers to $CSeq(S_1, 1) = (4, 4)$. In other words, it tells that, the first ordering of pattern $< a >$ in S_1 ends in IS_4, and this ordering has utility 4.

The root node, (ps:<>, u:0, CSeq: {}) is an empty pattern, while the leaves are the nodes that cannot grow any more or that are not allowed to grow due to pruning strategies applied. Item-based expanded nodes precede the itemset-based expanded nodes. For the same expansion technique, however, nodes are arranged in lexicographic order. The minimum utility threshold parameter, $minutil$, is used to prune the search space, and if it is set to 0, the complete search space is covered.

4.4 Calculating the Utilities of Child Nodes

The utilities of the child nodes are calculated by using the $CSeq$ of the parent. Expansion with an item results in a new node whose pattern structure is a revised version of that of the parent's. While utilizing $CSeq$, instead of traversing the whole database, only the sequences in $CSeq$ are examined for expansion. In addition, using the $last\_IS$ field, the essential subsequences of these transactions are examined instead of examining each sequence as a whole.

For item-based expansion, potential items to be added to the last itemset of the parent are the ones that are following the last item of the last itemset of the pattern lexicographically. In addition, from this set, only the items that follow the last item of the pattern which exist in the $last\_IS$ of the sequences of $CSeq$ are considered. We illustrate this by an example. Consider the pattern $< bd >$, the structure of which is given in Table 4.1. The pattern's last (and the only) itemset is *(bd)* and the last item is *d* in this itemset. For item-based expansion, the only possible item that can be added to the last itemset is *{e}*. For each ordering in $CSeq$, only item *{e}* in the itemset $last\_IS$ of the corresponding sequences are examined. Therefore, for the first ordering of S_2 in $CSeq$, $S_2(e, 2)$, and for the second ordering, $S_2(e, 3)$ are examined. For the orderings of S_5 and S_6, $S_5(e, 3)$ and $S_6(e, 2)$ are examined. HuspExt considers

Table 4.2: Pattern Structure for $< b(de) >$

ps			$< b(de) >$
u			18
CSeq			
Sequence	Ordering	$last\_IS$	u
S_2	1	3	18

only the ones that have *u* value greater than 0 for expansion, which is, $S_2(e, 3)$ for this case. Therefore, HuspExt has a single promising item for expansion and before candidate generation, it checks whether this item is really promising by running the pruning process. As seen in Figure 4.1, $S_2(e, 3)$ is *(u:12, ru:12, is:3)*. Currently, assume that, item *e* is promising. The item-based expansion of $< bd >$ with *e* results in pattern $< b(de) >$ whose utility is calculated using Equation 2. The resulting pattern structure for $< b(de) >$ is displayed in Table 4.2. Here, the sum and max functions are application dependent, and the solution supports any other function definition as well, which makes it a generic framework for high utility sequential pattern mining.

For itemset-based expansion, all items should be considered for expansion. For each sequence in $CSeq$, only the items in itemsets following the $last\_IS$ are the possible candidates. We illustrate the itemset-based expansion again for $< bd >$. For the first ordering of S_2 in $CSeq$, itemsets following $last\_IS$, which is IS_3, is examined. Considering the data matrix of S_2, items $S_2(b, 3)$, $S_2(d, 3)$ and $S_2(e, 3)$ have u values greater than 0. So possible items for expansion from this ordering are b, d, and e. Similar discussions hold for the other orderings. The utility calculation of candidate patterns is performed in the same manner with item-based expansion, after possible items for expansion are discovered.

4.5 Pruning Strategies

As described in the previous subsection, solution selects a subset of items for expansion using the $CSeq$ structure and the properties of the expansions. To further avoid selecting unpromising items, we propose a PBCG strategy, which is based on *CRoM*. Below, we provide the necessary definitions and the theorems before formal-

izing CRoM.

Definition 4.5.1. *RMUB (Rest of Match Based Upper Bound): Given a sequence S, $RMUB(P, S, i)$ is an upper bound on the utility of the candidate patterns with respect to S, created either by item-based or itemset-based expansion of a pattern P with item* i*, which is formulated as follows.*

For item-based expansion, if $S^{ru}(i, m)$ is greater than 0, then,

$$RMUB(P, S, i) = u(P, S) + S^{ru}(i, m) \tag{4.1}$$

otherwise $RMUB(P, S, i) = 0$.

For itemset-based expansion,

$$RMUB(P, S, i) = u(P, S) + S^{ru}(i, m + 1) \tag{4.2}$$

where $m = CSeq_P^{last\_IS}(S, 1)$.

Theorem 1. *Let* I *be a set of items, $i \in I$ be a candidate item for expansion, P be a pattern to be expanded, and S be a sequence from sequence database SD. For any pattern P' that is generated by item-based or itemset-based expansion of P with a **single item** i, has utility no more than $RMUB(P, S, i)$ in S, that is, $u(P', S) \leq RMUB(P, S, i)$.*

Proof. Let *k* and *k'* be the number of orderings of P and P' in S such that $k \geq k'$. Then,

$$\begin{aligned} u(P', S) &= max\{u(P', S, O_m) : m = 1, \ldots, k'\} \\ &= max\{u(P, S, O_j) + u(i, IS, S) : j = 1, \ldots, k\} \end{aligned} \tag{4.3}$$

where, in Equation (4.3), $u(i, IS, S)$ stands for the utilities of i in the itemsets following the last itemset of any of the orderings O_j of ***P***, such that, adding ***IS*** to O_j results in one of the orderings O_m of ***P'***.

Since $S^{ru}(i, IS) \geq u(i, IS, S)$, we can rewrite Equation (4.3) as follows.

$$u(P', S) \leq max\{u(P, S, O_j) + S^{ru}(i, IS) : j = 1, \ldots, k\} \tag{4.4}$$

By definition, ru value of any item in the $last\_IS$ of the first ordering of P is always greater than any other ordering's, that is, $S^{ru}(i, IS) \leq S^{ru}(i, m)$ where m is the

$last\_IS$ of the first ordering of P in S. This is due to the fact that, first ordering always ends in an itemset prior to any other ordering, therefore counts more items in its ru value. Therefore, for item-based expansion,

$$u(P', S) \leq max\{u(P, S, O_j) + S^{ru}(i, m) : j = 1, \ldots, k\} \tag{4.5}$$

and for itemset-based expansion,

$$u(P', S) \leq max\{u(P, S, O_j) + S^{ru}(i, m + 1) : j = 1, \ldots, k\} \tag{4.6}$$

since $S^{ru}(i, m)$ is independent of *j*, we can rewrite Equation (4.5) as follows.

$$\begin{aligned} u(P', S) &\leq max\{u(P, S, O_j) : j = 1, \ldots, k\} + S^{ru}(i, m) \\ &= u(P, S) + S^{ru}(i, m) = RMUB(P, S, i). \end{aligned} \tag{4.7}$$

So we conclude that,

$$u(P', S) \leq RMUB(P, S, i) \tag{4.8}$$

Similar steps are taken for itemset-based expansion after Equation (4.6). Thus, the theorem holds. □

In order to clarify the idea, consider for the pattern $< bd >$. Assume that, HuspExt is at a step in which it tries to find out an upper bound on the utility of the patterns that will be generated by expanding the pattern $< bd >$ with a single item *e* with respect to only sequence S_2. As it is defined in Theorem 1, $RMUB(< bd >, S_2, e)$ is an upper bound on the utility of these candidate patterns. For item-based expansion of $< bd >$ with item *e*, $m = CSeq^{last\_IS}_{<bd>}(S_2, 1)$=2. From Figure 4.1, $S^{ru}_2(e, IS_2) = 12$. Therefore, $RMUB(< bd >, S_2, e) = 7 + 12 = 19$.

For itemset-based expansion of $< bd >$ with item *e*;

$$\begin{aligned} RMUB(< bd >, S_2, e) =& u(< bd >, S_2) + \\ & S^{ru}_2(e, CSeq^{last\_IS}_{<bd>}(S_2, 1) + 1) \end{aligned} \tag{4.9}$$

In Equation (4.9), $CSeq^{last\_IS}_{<bd>}(S_2, 1) + 1 = 3$. Using Figure 4.1, $S^{ru}_2(e, IS_3) = 12$. Therefore, $RMUB(< bd >, S_2, e) = 7 + 12 = 19$.

For any expansion technique, a $RMUB$ value of 0 for a pattern P, item i and sequence S means that, the pattern generated by expanding P with i is not contained in sequence S. Therefore, the pattern does not have utility from that sequence.

Theorem 2. *Let I be a set of items, $i \in I$ be a candidate item for expansion, P be a pattern to be expanded, and S be a sequence from sequence database SD. Let P' be the generated pattern resulted from item-based or itemset-based expansion of P with* i. *Then the utilities of P' and its offsprings are no more than $RMUB(P, S, i)$ in S.*

Proof. The theorem has two parts that needs to be proved. First part is; *the utility of P' is no more than $RMUB(P, S, i)$ in S, that is, $u(P', S) \leq RMUB(P, S, i)$.* This statement is already proved by Theorem 1, since Theorem 1 states that single-item expansion of a pattern P results in a pattern P' whose utility is less than or equal to $RMUB(P, S, i)$.

The second part of the theorem states that *the utilities of none of the offsprings of P' has utility more than $RMUB(P', S, i)$ in S.* We will prove this by induction on the number of expansions, call it *x*, that are performed to get an offspring from P'. Using Theorem 1, the statement holds for $x = 1$.

Assume the statement holds for $x = d$, that is, all patterns generated by *d* number of expansions from P' *(call the set of these patterns as SP")*, have utility less than or equal to $RMUB(P, S, i)$ in S. All offsprings produced by $d + 1$ number of expansions from P' are generated by a single item expansion from the patterns in SP''. Since, by induction hypothesis, we know that all SP'' have utility less than or equal to $RMUB(P, S, i)$, one item expansion of patterns in SP'' should also have utility less than or equal to $RMUB(P, S, i)$ by Theorem 1. Therefore, the statement also holds for $x = d + 1$ and the proof of the induction step is complete. □

In order to better understand the statement behind Theorem 2, consider the expansion of pattern $< bd >$ with item *e* with respect to sequence S_2. As we calculated above, $RMUB(< bd >, S_2, e)$ is *19* for both item-based and itemset-based expansions. With Theorem 2, we can conclude that, the utility of patterns $< b(de) >$, $< bde >$ and none of their offsprings can be greater than 19 with respect to S_2. Considering the $RMUB$ value of the itemset-based expansion of $< bd >$ with item *e*, which is 19, we can conclude that, the upper bound on the utility value of pattern $< bde >$ and any of the patterns generated from it cannot exceed 19.

Definition 4.5.2. *CRoM (Cumulated Rest of Match) Based Upper Bound: $CRoM(P, i)$*

defines an upper bound on the utility of the pattern created by expanding P *with* i*, which is formulated as*

$$CRoM(P, i) = \sum_{S \in SD} RMUB(P, S, i) \tag{4.10}$$

Theorem 3. *Let* I *be a set of items,* $i \in I$ *be a candidate item for expansion,* P *be a pattern to be expanded, and* SD *be a utility based sequence database from* I*. Let* P' *be the generated pattern resulted from item-based or itemset-based expansion of* P *with* i*. Then the utilities of* P' *and its offsprings are no more than* $CRoM(P, i)$*.*

Proof. Let *P"* be any pattern generated by a number of expansions from P. Using Theorem 2,

$$\begin{aligned} u(P'') &= \sum_{S \in SD} u(P'', S) \\ &\leq \sum_{S \in SD} RMUB(P, S, i) = CRoM(P, i) \end{aligned} \tag{4.11}$$

□

Therefore, $CRoM$ defines an upper bound on the utilities of all the patterns generated from P', including P' itself.

In order to exemplify the calculation of $CRoM$ and the idea behind, consider again the expansion of pattern $< bd >$ with item *e*.

$$CRoM(< bd >, e) = \sum_{S \in SD} RMUB(< bd >, S, e) \tag{4.12}$$

Since the *CSeq* of $< bd >$ includes entries for S_2, S_5 and S_6,

$$\begin{aligned} CRoM(< bd >, e) = & RMUB(< bd >, S_2, e) + \\ & RMUB(< bd >, S_5, e) + \\ & RMUB(< bd >, S_6, e) \end{aligned} \tag{4.13}$$

For both types of expansions, $RMUB(< bd >, S_5, e)$ and $RMUB(< bd >, S_6, e)$ are equal to 0, and therefore, $CRoM(< bd >, e)$ equals to 19. Therefore, for this example, we can conclude that, utilities of $< b(de) >$, $< bde >$ and none of their offsprings can exceed 19.

$CRoM$ is used to eliminate promising items prior to candidate generation. More specifically, an item *i* is selected to be used in expansion of pattern P, if the $CRoM$

of the resulting pattern for that item is greater than or equal to the utility threshold. Considering the calculation complexity, to test whether an item is promising, HuspExt simply adds already calculated utility of the pattern $u(P, S)$ with ru values, which are calculated only once during the construction of data matrices of sequences. Therefore, the calculation process simply scans the sequences that exist in $CSeq$ only once.

We claim that, with CRoM, we can eliminate more promising items prior to pattern generation, in comparison with the utilization of SDCP. We prove our claim with Theorem 4.

Theorem 4. *Given a utility based sequence database SD, and a pattern P for expansion, the number of promising items eliminated by SDCP is less than or equal to the number of items eliminated by CRoM.*

Proof. We will prove the theorem by contradiction. Suppose that the number of items eliminated by SDCP is greater than the number of items eliminated by CRoM. Then, for a candidate pattern P, there should at least be one item *i* such that, expansion of P with *i* results in P' for which $SWU(P') < CRoM(P', i)$ holds.

Let SD' be the set of sequences from SD that contains P'. For any $S \in SD'$, let $R(S, i)$ be the right subsequence of S following item i, including *i* itself, for the ordering that has maximum utility value for P' in S. Then,

$$\begin{aligned} SWU(P') &= \sum_{S \in SD'} u(S) \\ &\geq \sum_{S \in SD'} (u(P, S) + u(R(S, i)) \\ &= \sum_{S \in SD'} (u(P, S) + S^{ru}(i, IS)) \end{aligned} \tag{4.14}$$

where *IS* is the itemset of S that contains the newly added item *i*, if *i* is to be used in item-based expansion; otherwise, *IS* is the following itemset. Therefore,

$$\begin{aligned} SWU(P') &\geq \sum_{S \in SD'} RMUB(P, S, i) \\ &= CRoM(P, i) \end{aligned} \tag{4.15}$$

□

For instance, consider the pattern $< bde >$, which exists only in S_2 of the example database. $u(< bde >)$ was calculated as 19 and we have evaluated $CRoM(< bd >, e)$

as 19 also. From the definition of SWU [61], SWU($< bde >$) equals to the utility of S_2. Using Table 2.2 and 2.3, SWU($< bde >$)=27, which is higher than CRoM($< bde >$). We can conclude that, for any utility threshold value $minutil > 19$, CRoM will eliminate $< bde >$. However, $< bde >$ cannot be eliminated by SWU for $minutil \leq 27$. For instance, for the scenario where $minutil$ is set to 20, $< bde >$ is not a high utility sequential pattern, since $u(< bde >) = 19$ which is smaller than 20. CRoM can correctly eliminate $< bde >$. However, since SWU value of the pattern is 27, $< bde >$ is a potential candidate for SWU based technique, therefore, it cannot be eliminated.

The prediction using CRoM becomes more accurate as the lengths of the patterns increase. This is due to the fact, both CRoM and SDCP overestimate the utility of the patterns however, overestimation using CRoM decreases as the lengths of the patterns increase. Moreover, overestimation stays the same for SDCP at each step of the algorithm and at each level of the pattern tree. In addition, considering different datasets, the prediction using SDCP overestimates more, as the lengths of the extracted patterns decrease and the dataset contains sequences with longer lengths.

4.6 HuspExt Algorithm

The HuspExt algorithm is given in Algorithm 1. It includes two procedures. First procedure, $HuspExtMain$, is the entrance point of the solution (Lines 1-3) which takes a utility based sequence database SD and a minimum utility threshold $minutil$ as input. Initially, utility values are calculated using external utilities in the external utility table and internal utilities in the initial sequence database. SD includes these calculated utilities. It outputs all the high utility sequential patterns. $HuspExtMain$ creates the root node with zero utility and empty parent string (Line 2). Then, it calls the second procedure $ExpandNodes$ to extract all high utility sequential patterns (Line 3).

$ExpandNodes$ (Lines 4-16) handles the construction of the pattern tree. The input to the procedure is a parent pattern, $parentP$. As the first step, if $parentP$'s utility is not less than the utility threshold, it is outputted as a high utility pattern (Lines 5-6).

Algorithm 1 HuspExt Algorithm

1: **procedure** HUSPEXTMAIN($SD, minutil$)

Input: utility-based sequence database SD, minimum utility $minutil$.

Output: all sequential patterns *p* that have $u \geq minutil$.

2: $root \leftarrow$ create new pattern (pattern string:<>, utility:0, CSeq:{})

3: EXPANDNODE($root$)

4: **procedure** EXPANDNODE($parentP, SD, minutil$)

Input: pattern parentP, utility-based sequence database SD, min.utility $minutil$.

Output: all sequential patterns *p* that have $u \geq minutil$.

5: **if** parentP.utility $\geq minutil$ **then**

6: output parentP as high utility pattern

7: $iList \leftarrow$ promising items for item-based expansion from parentP

8: **for all** $i \in iList$ **do**

9: **if** $CRoM(parentP, i) \geq minutil$ **then**

10: $childP \leftarrow$ create new pattern by item-based expanding parentP with i

11: EXPANDNODE($childP$)

12: $itList \leftarrow$ promising items for itemset-based expansion from parentP

13: **for all** $i \in itList$ **do**

14: **if** $CRoM(parentP, i) \geq minutil$ **then**

15: $childP \leftarrow$ new pattern by item-based expanding p with i

16: EXPANDNODE($childP$)

Lines 7-11 and 12-16 are the item-based and itemset-based expansions of $parentP$, respectively. For each expansion technique, first the promising items are collected that are suitable for each type of expansion (Lines 7, 12) as described in Section 4. Collected items are pruned before candidate generation using the $CRoM$ values with respect to $parentP$ as calculated by Equation (12) (Lines 9, 14). If the $CRoM$ values are greater than or equal to $minutil$, then child patterns of the $parentP$ are formed with the proper structure using the structure of $parentP$. $ExpandNodes$ recursively invokes itself with the newly created child nodes to go deeper in the pattern tree (Lines 11, 16).

When the computational complexity is considered, a single scan of the database is performed for each call of both of the procedures so as to find promising items and calculate the $CRoM$ values. More specifically, only $HuspExtMain$ performs a whole database scan. $ExpandNodes$, for each call, scans only the sequences in $CSeq$ of parent pattern, and it scans only the itemsets following the last itemset of the corresponding sequence that match $parentP$, including the last itemset. At a single scan, it finds all promising items, and calculate the $CRoM$ values. Since for each item i, $CRoM$ is the cumulated $S^{ru}(i, m+1)$, where m is the last itemset of each sequence S in $CSeq$ of $parentP$, its overhead is just a direct access of a data matrix, which is constant $O(1)$.

For the memory consumption, it is important to note that, we use conventional sparse matrix solutions for database sequences, such that, we do not use extra memory storage for the items that do not exist in a sequence. As for the patterns, CSeq has the highest memory consumption. It contains information about observed orderings, and the number of orderings generally decreases as the length of the patterns increases. We can give a general upper bound for memory consumption as $O(n \times k!)$ for n sequences each with maximum k itemsets. However, this is a highly overestimated upper bound since considering the real datasets, having all $k!$ occurrences of a pattern in a sequence is not practically observed.

CHAPTER 5

SEQUENTIAL PATTERN EXTRACTION UNDER USER-DEFINED PATTERN SCORING

In this chapter, proposed solution for sequential pattern extraction under user-defined pattern scoring is described. The solution is designed as a generic framework, and in this framework, user provides a function for pattern evaluation such that, this function can be considered as a module which can be replaced with another function in order to satify another user-defined scoring.

The current version of the technique is evaluated in the web usage domain, where a web access pattern evaluation function is designed. The initial version of the solution is presented in [9], which is enhanced by a new pattern extraction approach and extensive evaluation of the framework by well-known datasets from literature in the current solution. The proposed solution is a hybrid framework in the sense that it combines clustering with a search-based pattern extraction algorithm. The solution is based on a two-step process for extracting patterns. In the first step, the database sequences are clustered. Next, in the second step, patterns are extracted from each cluster by using a new algorithm, *WaPUPS*.

In the following subsections, the solution is presented by giving details about the data preparation and clustering phases, and the proposed algorithm WaPUPS for scoring based sequential pattern extraction.

5.1 Data preparation

The current version of the solution is adapted to work on the web usage data. However, it should be mentioned that, the developed framework can work with any other sequence data, as well.

Data preparation is the initial step and a critical point for every web usage mining solution. The tasks completed in this phase include *data cleaning, user identification, session identification, visit duration calculation, and session elimination*. In the experimental part of this research which is described in Chapter 6, we have used five different datasets. For the data sets which have not been already cleaned and sessionized, the same preprocessing steps are applied.

In the data cleaning part, the irrelevant log entries such as erroneous accesses, requests with the filename suffixes including *gif, jpeg, and jpg* are eliminated. In addition, there are web pages that are represented by syntactically different URLs in the log files. Such URLs are determined and normalized.

Several heuristics exist for identifying users and sessions from the web log data[18]. In our solution, we use a heuristic method that identifies a user by using the IP address and the browser information. For the session identification, a new session is created when a new user is encountered or if the page visit duration exceeds *30* minutes for the same user [17]. During the pattern extraction step, we consider each user's session as a sequence and the aim of our solution is to extract patterns satisfying a user-defined evaluation criteria.

Page visit duration, which is defined as the time difference between consecutive page requests, is used in the pattern extraction phase while evaluating the patterns. The calculation of the visit duration is straightforward for accesses except the last page in a session. For the last page, visit duration is set to the mean of the times for that page taken across all the sessions in which it is not the last page request. As the last step of the page visit duration calculation, the times are normalized across the visiting durations of the pages in the same session, such that, the normalized time has a value between *1* and *10*. This normalization process captures the relative importance of a page to a user in a session.

The last step that is performed before pattern extraction is the session elimination, in which sessions that have length less than *5* are removed in order to eliminate the effect of random accesses to the web site.

5.2 Clustering Sequences

In order to discover distinct behavior groups, sequences are first clustered into groups of similar interests. Partitioning based clustering techniques, specifically, *k-means algorithm*, have been proven to be efficient for identifying the intrinsic common attributes in web log data [44], and it is efficient on large datasets due to its linear time complexity. Hence, k-means clustering is utilized at this step.

Clustering is performed on session-access matrix, where each column is an access and each row is a session of any user represented as a vector. The matrix values are the frequency of the corresponding web page's visits within that session.

5.3 WaPUPS algorithm

Each user session in a cluster is a sequence of web pages visited by a single user. For each cluster, access patterns are built by using a search oriented approach. Different from previous web usage mining techniques, usage patterns are discovered by considering the evaluation criteria defined dynamically to select the most representative page access and to include it into the pattern. In order to realize this, we have devised the *WaPUPS* algorithm.

WaPUPS algorithm constructs patterns in a recursive manner. The algorithm can be considered as the construction of a tree where at each recursive call, one level of the tree is built, which corresponds to adding a new access to the parent pattern. At each level of the tree, at most *Branching Factor (BF)* number of nodes exists.

The algorithm is presented in Algorithm 2. WaPUPS takes a list of *BF* number of parents and cluster number from which the patterns are extracted, as input. The output of the algorithm is the set of extracted patterns of that cluster. For each recursive call,

Algorithm 2 WaPUPS Algorithm

Input:

$parent\_list$: list of parent states that will create children at the next iteration, has *BF* elements

cl_no: cluster whose sessions are used during building patterns

Output:

$WAPList$: extracted web access patterns

1: $Access_List \leftarrow$list of all possible accesses in the dataset
2: $child\_array \leftarrow$ create an empty array that has BF elements, sorted in descending order of value
3: **for each** $parent\_pattern \in parent\_list$ **do**
4: **for each** $access \in acccess\_list$ **do**
5: $new\_pattern \leftarrow$ EXTEND$(parent_pattern, access)$
6: $new\_pattern.value \leftarrow$ EVALUATE$(new_pattern, cl_no)$
7: **if** $new\_pattern.value > 0$ **then**
8: add $new\_pattern$ to $child\_array$
9: increment $parent\_pattern.child\_count$
10: **for each** $parent\_pattern \in parent\_list$ **do**
11: **if** $parent\_pattern.child\_count = 0$ **then**
12: add $parent\_pattern$ to $WAPList$
13: **if** $child\_array$ is not empty **then**
14: WAPUPS$(child_array, cl_no)$

the *parent_list* keeps the patterns that have been constructed up to the current call. These are the best *BF* patterns of the previous call which are kept in the *child_array*. Therefore, *child_array* always keeps *BF* patterns with the highest evaluation values that will be the parent nodes of the next recursive call.

The construction of the *child_array*, which will become the parameter of the next recursive call is completed with the first for-loop in Algorithm 2, when all the parent nodes are checked with all possible requests to get enhanced with a new access (Lines 3-9). As mentioned before, in the second for-loop (Lines 10-12), if any of the parent nodes in the *parent_list* cannot grow, it is outputted as a pattern for that cluster. Finally, if the *child_array* contains at least one member, the function is recursively called with the *child_array* as the *parent_list* for that cluster (Lines 13-14).

5.4 Evaluation Function

The Evaluate function examines a pattern and assigns a value to it. In this work, in order to demonstrate the proposed solution and the evaluation function, we consider access patterns as paths that will be recommended to web users for faster web site traversals and we design a sample evaluation function accordingly. We aim to assign values to patterns considering the degree they are assumed to facilitate the users' navigation in the web sites. In this sample evaluation function, the value of a pattern depends on the following factors:

1. the number of sequences in cluster *cl_no* in which the pattern exists;
2. the number of distinct users that traverse the pattern in their sessions;
3. the total duration of the pattern in sequences (calculated as the sum of the page visit durations of all the accesses from the start access of the pattern to the final access).

We assume that, the value of a pattern increases as the number of sessions it exists in increases. However, it is possible that these sessions belong to a small number of users who access the web page frequently. Therefore, our assumption is based on the

Algorithm 3 Evaluation Function Used in WaPUPS Algorithm

Input:

$new\_pattern$: pattern to be evaluated,

cl_no: cluster number for which web access patterns will be constructed.

Output:

$value$: value of the pattern.

1: $NumberOfSeq \leftarrow$ number of sequences in cluster *cl_no*

2: $NumberOfContSeq \leftarrow$ number of sequences in cluster *cl_no* that contains *new_pattern*

3: $TotalDistinctUsers \leftarrow$ number of distinct users that own sequences in cluster *cl_no*

4: $Distinct\_User\_List \leftarrow$ distinct users' list that followed *new_pattern* in sequences

5: $containing\_sequences \leftarrow GetAllSequences(new\_pattern, cl\_no)$ ▷ *get all sequences in cl_no that contains new_pattern*

6: $duration \leftarrow 0$

7: **for each** $sequence \in containing\_sequences$ **do**

8: $user \leftarrow sequence.user$

9: **if** $user \notin Distinct\_User\_List$ **then**

10: ADDUSER($Distinct\_User\_List, user$)

11: $duration \leftarrow duration+$ time to traverse $new\_pattern$ in sequence

12: $value \leftarrow$

$(NumberOfContSeq/NumberOfSeq) \times w\_session\_count+$

$(Distinct\_User\_List.size/TotalDistinctUsers) \times w\_user\_count+$

$(duration/NumberOfSeq) \times w\_duration$

idea that, a pattern that claims to shorten long access sequences of web users should be useful for as many users as possible. Hence it should exist in many distinct users' access sequences. In addition, the aim is to shorten long access sequences, therefore, the value of a pattern increases as the length of the path it shortens increases. The evaluation algorithm that realizes these ideas is given in Algorithm 3. Although the current evaluation function uses these values, it can be modified to depend on other factors as well.

As it is shown in Algorithm 3, the algorithm finds all the dataset sequences that contain the pattern to be evaluated (Line 5). These sequences are examined in order to find out how many distinct users own them and the total duration of the pattern in them (Lines 7-11). The value of the pattern is finally evaluated by normalizing these three values and weighting them with the three parameters of the solution: $w_session_count$, w_user_count and $w_duration$ (Line 12). We have evaluated different values of the weight parameters in this formula, however, it should be mentioned that, the weights can be adjusted for different web site configurations, and the formula can be modified so as to take any information available to the web site into account.

In order to better understand the idea of the evaluation, consider the following simple example: assume that we have a *pattern* (A, B, C) to evaluate, and 3 user sessions where this pattern exists {user1,(D-F-A-H-J-H-I-B-F-C-P)}, {user2,(A-B-F-U-C)}, and {user1,(T-A-F-H-B-C-A-D)}. Assume further that, page visit duration for each access in all of the sessions is 1, to keep the example simple. Then the number of sessions that this pattern exists in is 3. The number of distinct users that covers this pattern in their access sequences is 2, and the total duration of this path in sessions is 18, due to the sum 8 (A-H-J-H-I-B-F-C) + 5 (A-B-F-U-C) + 5 (A-F-H-B-C). We then calculate the value of this pattern using the formula given in line 12 of Algorithm 3. As it is shown in Chapter 6, we have evaluated different values of the weight parameters in this formula, however, it should be mentioned that, the weights can be adjusted for different datasets, and the formula can be modified so as to take any information available to the target users into account.

CHAPTER 6

EVALUATION OF THE PROPOSED SOLUTIONS

In this chapter, details of the experimental evaluation of the proposed solutions are described. For both of the solutions, the properties of the evaluation environment, the datasets used during the experiments, evaluation methodology and results are given. Discussion of the results and comparison with the state of art techniques are provided.

6.1 Evaluation of the Proposed Solution for High Utility Sequential Pattern Mining

In this section, evaluation of the HuspExt algorithm that uses CRoM based pruning is presented. HuspExt was implemented in Java and developed in Oracle JDeveloper. All the experiments were performed on a 3.3 GHz Intel Processor with 8 GB main memory on Windows 7 operating system. In order to evaluate the proposed solution in an exhaustive manner, we made experiments for determining the strength of the solution from different aspects including time and memory requirements as well as the number of patterns that are generated during execution. Additively, experiments are performed under datasets from different domains so as to understand the behavior of the solution under different dataset characteristics.

In order to evaluate the pruning power of HuspExt, which is based on CRoM, an SWU based algorithm, called *SWU* is implemented using the same development environment and data structures as in HuspExt. This enables us to make fair comparison of the pruning strategies. During implementation of *SWU*, we modified HuspExt such that, it keeps the sum of the utilities of all the items in a sequence, that is the *se-*

Table 6.1: Evaluation Phases

Phase	Purpose	Datasets	Methods
Phase 1	Determine the characteristics of the datasets in terms of the number of patterns, the length of the patterns, the feasible minimum utility thresholds that HuspExt and compared solutions can produce results	CENG, Foodmart, Brightkite, DS3, ChainStore, D100kT3S6N1k, D100kT6S4N10k	HuspExt, SWU, USPAN
Phase 2	Performance Comparison with Existing Studies	CENG, Foodmart, Brightkite, DS3, D100kT3S6N1k, D100kT6S4N10k	HuspExt, USPAN
Phase 3	Evaluation of the proposed PBCG strategy	CENG, Foodmart, Brightkite, DS3, D100kT3S6N1k, D100kT6S4N10k	HuspExt, SWU
Phase 4	Evaluation of the scalability of the solution under varying dataset sizes	D50kT3S6NXk, DXkT3S6N1k, ChainStore	HuspExt, SWU

quence utility, in *ru* field of an item in a sequence instead of the rest of the sum of the utilities following the corresponding item. In addition, for PBCG, SWU checks the SWU value of a pattern instead of RMUB. Using the same development environment enables us to make fair comparison of the pruning strategies.

In addition, we made experiments with the USpan algorithm. The authors of [61]

provided the executable file of their solution together with the dataset, named DS3, that was mentioned in the paper. The solution produces a result file which includes the time and space consumption of the algorithm. We have used DS3 in our experiments which is a real world dataset. Our evaluation phase can be divided into four phases, details of which are provided in Table 6.1.

Table 6.2: Properties of Real World Datasets

Dataset	Domain	#Seq.	Avg.Seq.Len.	#Items
CENG	Web access	5032	18.215	1500
Foodmart	marketing	5581	15.559	1560
Brightkite	location-based social networking	20,878	5.608	1500
DS3	shopping	59,477	5	811
Chain-Store	shopping	1,112,949	7.3	46,086

6.1.1 DataSets

Both real and synthetic datasets are used during evaluation. CENG, Foodmart, Brightkite, DS3 and Chain-Store are the real world datasets. CENG dataset[1] is from the Computer Engineering (CENG) Department of Middle East Technical University. The dataset contains web server logs from 03.09.2011 to 13.11.2011. After preprocessing, there remains 852,771 accesses, 5032 users and 9876 sessions. For the web domain, we adapted the following configuration; pageviews correspond to items, sessions correspond to itemsets and ordered sessions of host corresponds to a sequence.

[1] www.ceng.metu.edu.tr

For the experiments, we worked with the top 1500 accessed page views.

Foodmart is acquired from Microsoft foodmart 2000 database, and we use foodmart data of Sales Fact 1998. The dataset already includes external and internal utilities. In the dataset, every sale is associated with a product, customer, time, cost and unit. In the experimental configuration, products correspond to items, and utilities are evaluated by multiplying cost and unit values. The sales of a customer correspond to a sequence which is composed of an ordered list of itemsets that are grouped and ordered by the time of the customer's sales. Brightkite dataset[2]is a location-based social networking data set, where users shared their locations by checking-in. There is a total of 4,491,143 checkins of these users over the period of Apr. 2008 - Oct. 2010 from 58,228 distinct users. We used the time and location information of the users. For this dataset, locations correspond to items. Each user's locations in a day constructs an itemset of the sequence. For the experiments, we worked with the top 1500 checked-in locations.

Table 6.3: Parameter Settings of Synthetic Data Sets

Parameter	Description	Default
$\|D\|$	Number of sequences	100k
$\|T\|$	Average items per itemset	2.5
$\|S\|$	Average itemsets in maximum sequences	10
$\|N\|$	Number of distinct items	10k

DS3 is a real dataset consisting of online shopping transactions of customers. The dataset we delivered includes utilities. There are 811 distinct products, 350,241 transactions and 59,477 customers in the dataset.

Chain-Store is a real dataset consisting of shopping transactions obtained from NU-MineBench 2.0 [48]. The dataset contains unit profits and purchased quantities that are used to find item utilities. Each customer transaction is regarded as a database sequence. Chain-Store is a sparse dataset with many distinct items and it is therefore used in the scalability evaluation of the proposed solution. The properties of the real

[2] http://snap.stanford.edu/data/loc-brightkite.html

world datasets are summarized in Table 6.2.

Table 6.4: Real Dataset Characteristics

ϵ	#Patterns	Avg. Pat. Len.	Max.Pat. Len.
CENG Dataset			
0.00041	1200276	12.131	20
0.00043	802777	11.874	20
0.00045	553304	11.428	20
0.00047	399284	10.852	20
0.00049	305269	10.265	20
Foodmart			
0.00001	2394784	6.082	17
0.00003	2176764	6.404	17
0.00005	1708211	7.072	17
0.00007	1159796	7.873	17
0.00009	659933	8.710	17
Brightkite			
0.0001	136035	5.777	14
0.0003	135473	5.792	14
0.0005	133913	5.830	14
0.0007	130508	5.912	14
0.0009	125070	6.035	14
DS3			
0.00034	192357	28.894	36
0.00035	7819	1.948	5
0.00036	7285	1.935	5
0.00037	6878	1.930	3
0.00038	6448	1.926	3
Chain-Store			
0.00004	170313	9.046	18
0.00006	18561	4.907	18
0.00008	8050	2.101	15
0.00010	5522	1.911	10
0.00012	4120	1.870	5

Synthetic datasets are generated using IBM data generator [65]. D100kT3S6N1k, and D100kT6S4N10k are the two generated datasets that are used in the evaluation of the performance of the pruning strategies and the solution. The names of the datasets

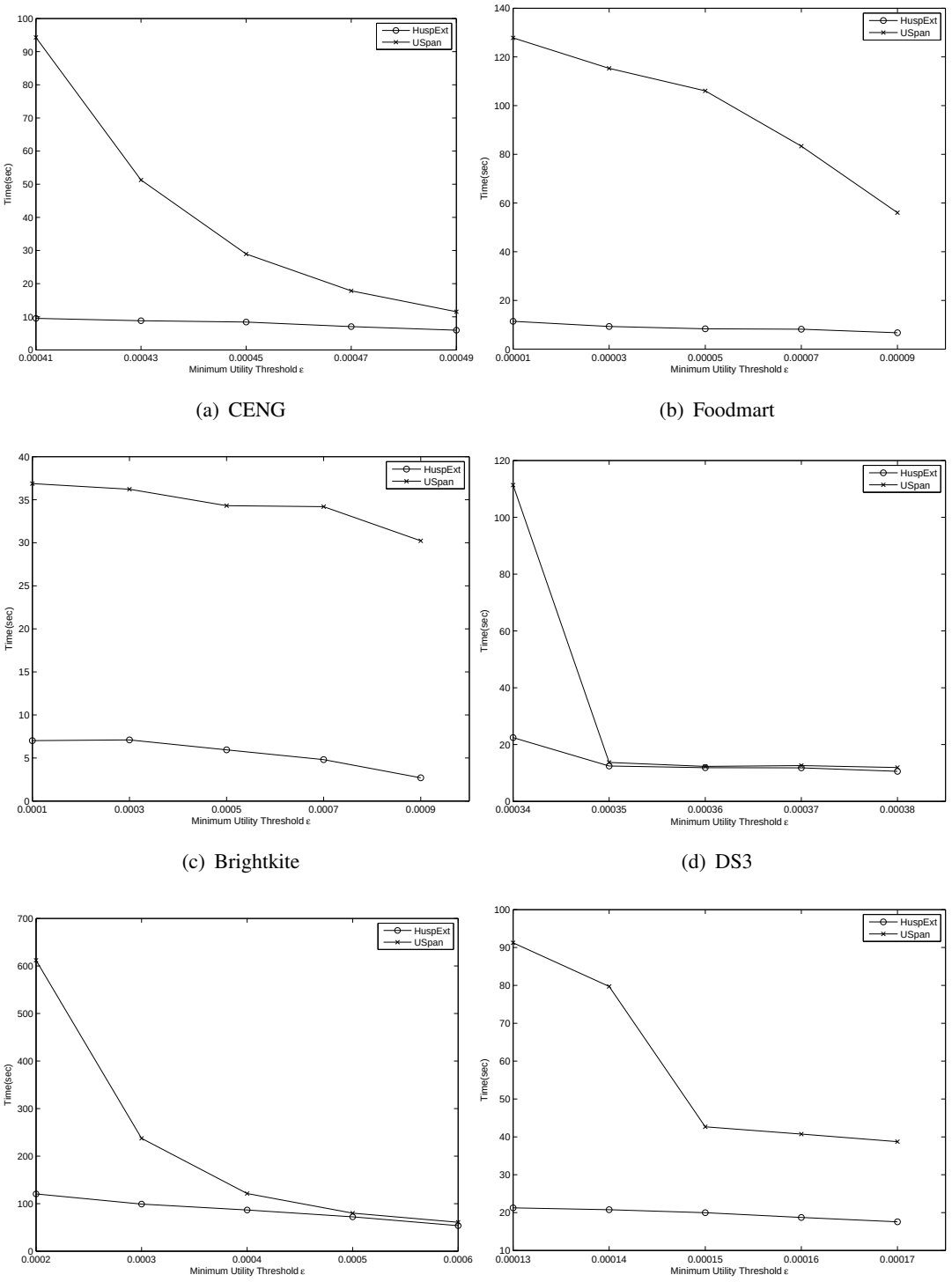
describe their properties in terms of the parameters, such that, each parameter has the value that follows it. The parameters correspond to the following values; *D* is the number of sequences, *T* is the average transaction length, *S* is the average length of a maximal pattern, and *N* is the number of items. In addition, ten more datasets are generated for the evaluation of the scalability of the solution under varying database volumes. For the datasets named DXkT3S6N1k, all parameters are the same except the number of transactions, *X*, which is gradually increased from 200 to 400 by 50. For the datasets, D50kT3S6NXk, only the number of items is changed from $2k$ to $10k$, increased by $2k$. Parameter descriptions and their default values are shown in Table 6.3.

Table 6.5: Synthetic Dataset Characteristics

ϵ	#Patterns	Avg. Pat. Len.	Max.Pat. Len.
D100kT3S6N1k			
0.0002	288276	6.646	16
0.0003	56479	6.312	15
0.0004	12104	5.800	14
0.0005	3004	4.333	12
0.0006	976	2.906	11
D100kT6S4N10k			
0.00013	35731	6.730	63
0.00014	29996	3.384	10
0.00015	26987	3.376	10
0.00016	24329	3.371	10
0.00017	21935	3.369	9

6.1.2 Phase 1: Dataset Characteristics

In this phase, we aim to discover the characteristics of the datasets and determine the proper utility threshold values that will be used in the following phases. For all of the experiments, thresholds are chosen in an unbiased manner such that, both the proposed algorithm and the compared techniques can respond within the computational limits of the evaluation environment. In all of the datasets, except for the Foodmart, utilities of the items are generated between 0.1 and 10 by using a log-normal distri-



(a) CENG (b) Foodmart

(c) Brightkite (d) DS3

(e) D100kT3S6N1k (f) D100kT6S4N10k

Figure 6.1: Performance comparison on real and synthetic datasets in terms of Time Complexity

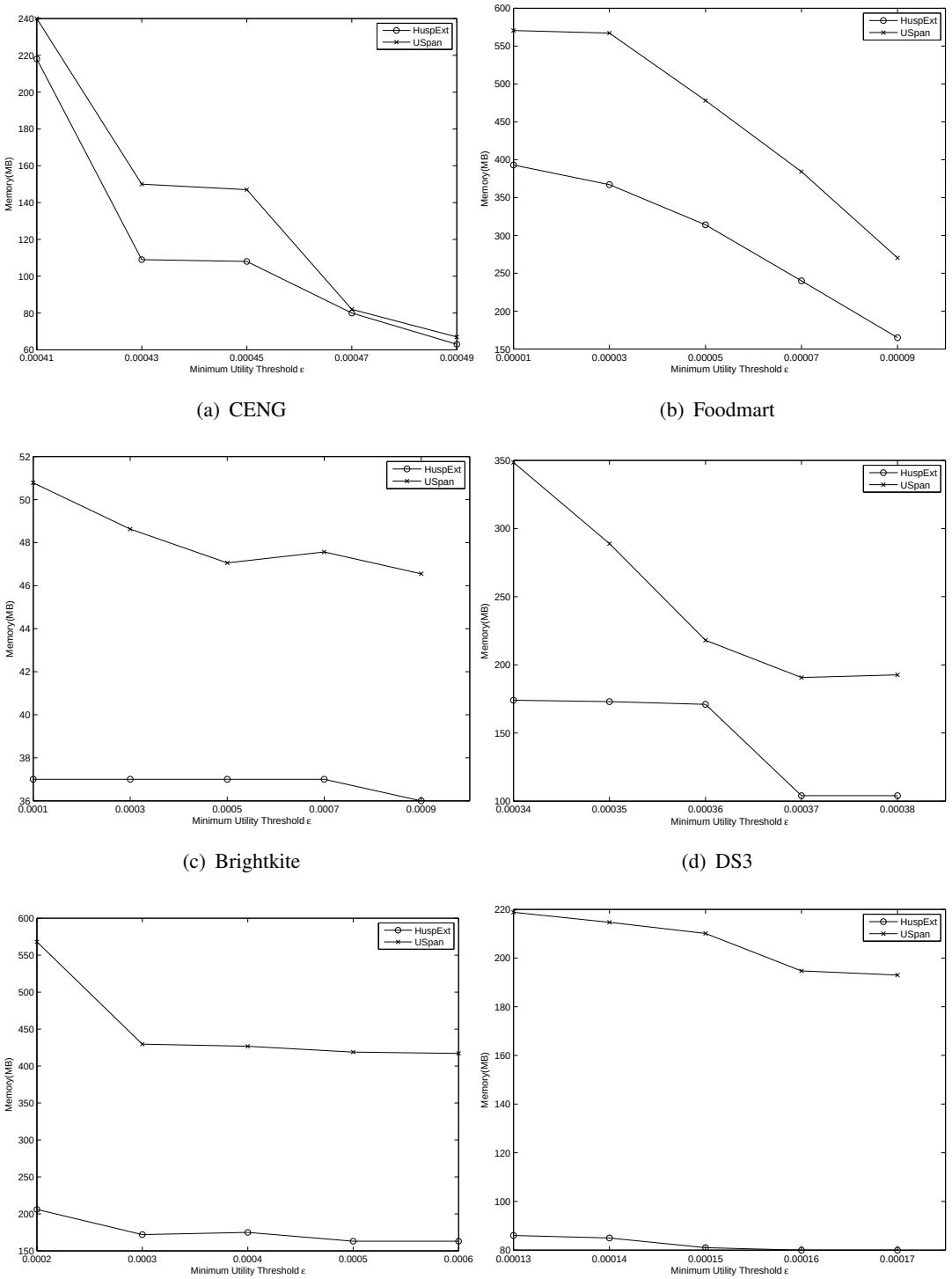
bution. In Tables 6.4 and 6.5, properties of the real and synthetic datasets are given, respectively. In these tables, first column lists the ϵ values, which are the minimum utility thresholds corresponding to a relative percentage of the utility of the sequence database, as described in Chapter 2.

6.1.3 Phase 2: Performance Evaluation

In this phase of the evaluation, we compare the performance of HuspExt with USpan, in terms of time and memory consumption. The experiments include the evaluation of the compared solutions under the thresholds given in Table 6.4. Figures 6.1 and 6.2 show the results. For all the datasets, the time consumed by HuspExt and USpan decreases as the threshold increases. Although the difference between the total runtime of USpan and HuspExt is large, small decrements cannot be seen in the figure when both of the solutions results are shown in the same graph.

The decrease in total runtime with increasing threshold can be explained by the fact that, with smaller threshold values, a deeper and wider search can be performed which results in identifying more patterns with longer lengths. This can also be observed in Table 6.4 where the maximum length of patterns and the number of extracted patterns increase as thresholds decrease. However, the increase in the total runtime for HuspExt is not as much as that of USpan. This can be explained by the characteristics of the dataset and the pruning performed by the algorithms. First of all, if the dataset has a significant number of patterns that have utility around some specific value, call it $minutil_1$, then the algorithm may respond for utility thresholds $minutil > minutil_1$ in short time. When $minutil \leq minutil_1$, there will be a sharp increase in response time due to a significant expansion in the search space, since for thresholds around $minutil_1$, there will be many patterns that can not be eliminated by a smaller threshold value. This observation can be supported by Table 6.4 and Figure 6.1 together.

When we analyze number of extracted patterns in Table 6.4, DS3 and D100kT3S6N1k are the two datasets that have a sharp increase in the number of extracted patterns after the threshold values 0.00034 and 0.0002 respectively. USpan's total runtime has a sharp increase for these two thresholds as well. HuspExt gets effected from these thresholds less than USpan, since it uses a tighter upper bound for the pruning pro-



(a) CENG (b) Foodmart

(c) Brightkite (d) DS3

(e) D100kT3S6N1k (f) D100kT6S4N10k

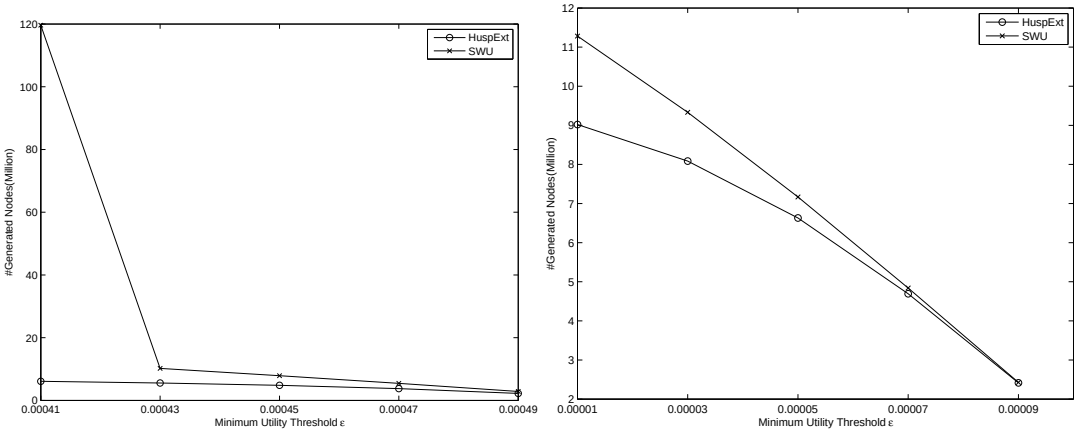
Figure 6.2: Performance comparison on real and synthetic datasets in terms of Memory Consumption

cess. Therefore, for instance, even for $minutil \leq minutil_1$, it can still eliminate unpromising candidates.

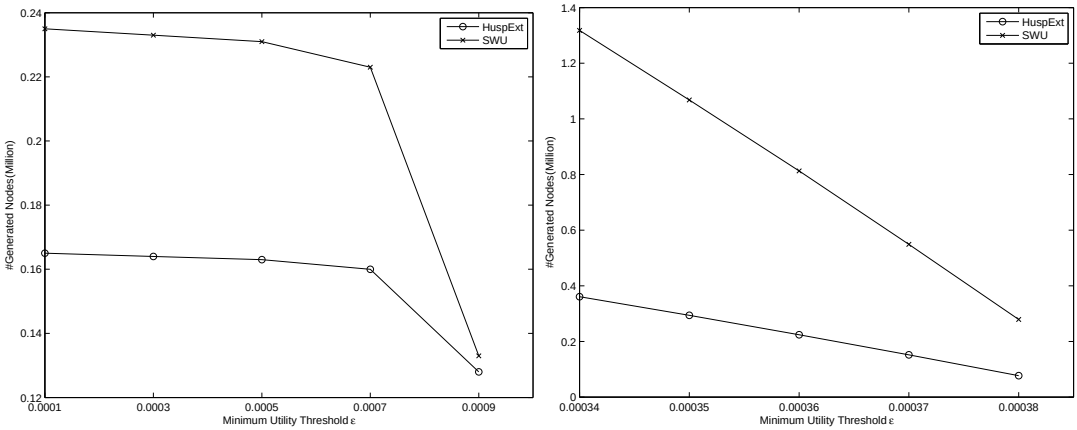
In addition to the above discussions, it can be observed that for all of the datasets, HuspExt has a significantly lower execution time and memory requirement than USpan. Especially for lower threshold values, the performance difference is higher since, HuspExt uses a tighter upper bound on the utilities of candidate patterns which results in generating less candidates, that leads to lower running times and memory consumption values.

6.1.4 Phase 3: Evaluation of the Proposed Pruning Strategy

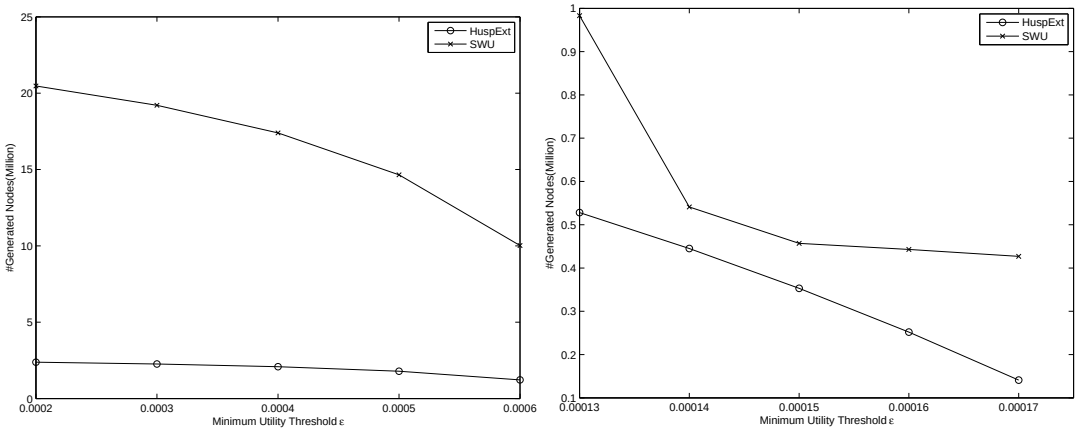
The main contribution of the proposed solution is the CRoM based pruning technique that is utilized for PBCG. With the pruning strategy developed, our aim is to eliminate more candidate patterns by using a tighter upper bound, in comparison with the state of art SWU based solution. To evaluate our pruning strategy, we compare our solution with SWU, in terms of the number of generated candidate patterns under varying minimum utility thresholds. Results of this evalution phase for different data sets are displayed in Figure 6.3. As seen in the figure, for all the datasets, HuspExt outperforms SWU. The difference between the number of the generated candidates is more for lower utility threshold values. This is due to the fact that, for higher threshold values, it becomes harder to eliminate candidates for both of the solutions.



(a) CENG (b) Foodmart

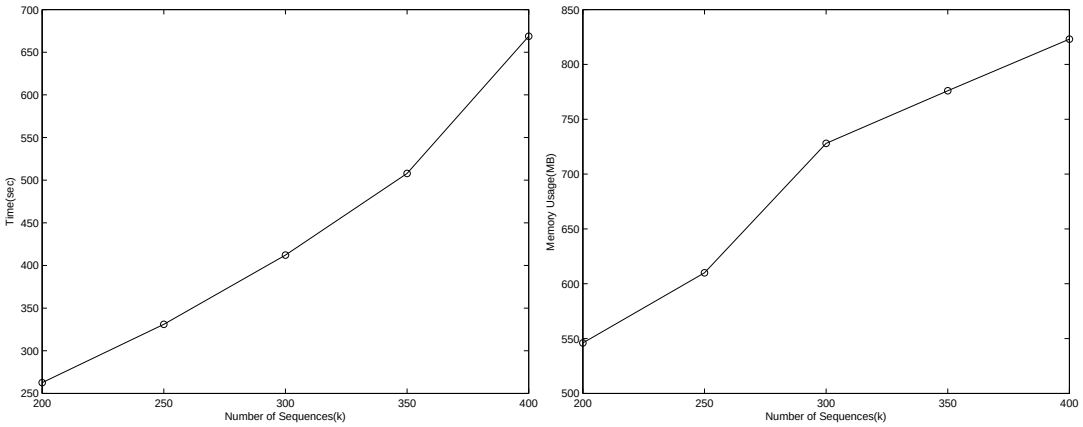


(c) Brightkite (d) DS3



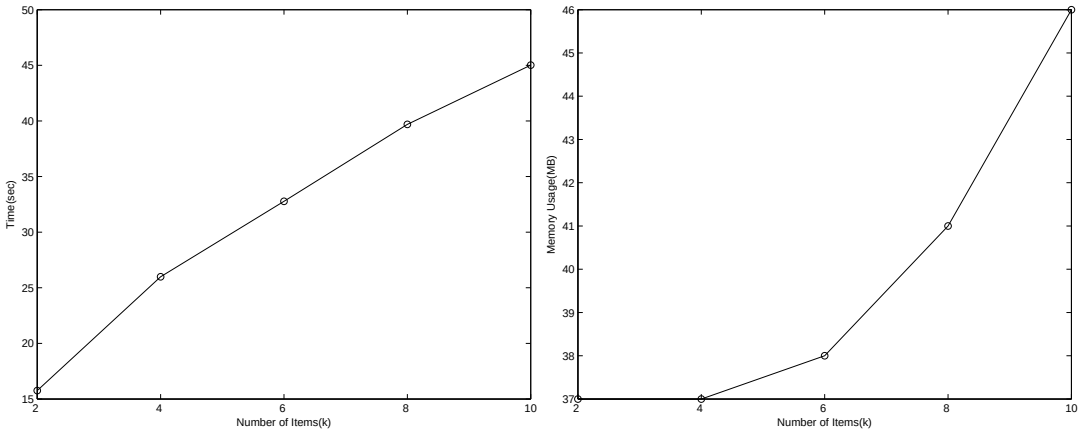
(e) D100kT3S6N1k (f) D100kT6S4N10k

Figure 6.3: Number of Candidates Generated During Execution



(a) Total Runtime (b) Memory Usage

Figure 6.4: Performance Analysis under changing Database Sizes *(DXkT3S6N1k)*



(a) Total Runtime (b) Memory Usage

Figure 6.5: Performance Analysis under changing Number of Distinct Items *(D50kT3S6NXk)*

6.1.5 Phase 4: Scalability of the Proposed Solution

We conduct scalability evaluation for the proposed solution under varying database sizes and distinct number of items. In addition, in this phase, we performed tests with a sparse dataset, Chain-Store, which is a real dataset with high number of transactions and distinct items.

For the evaluation of the solution under different data characteristics, we generated 10 synthetic datasets, abbreviated as D100kT3S6N1k and D100kT6S4N10k and set

utility threshold to 0.0001. Figures 6.4 and 6.5 display the results of the experiments in terms of execution time and memory usage. As it can be observed, both the running time and memory consumption of the solution increase as the size of the dataset and the number of distinct items increase. This increase is nearly linear with the increase in the number of sequences and items. As the number of database sequences increase, HuspExt has to traverse more sequences which increases the total runtime. In addition, HuspExt stores the whole dataset which increases the memory requirement also. Similarly, more distinct items in the database results in increase in the number of promising items, and hence the number of candidate patterns. This increases both the total runtime of HuspExt, and the memory it needs to store the candidates.

Table 6.6: Scalability Evaluation with Chain-Store Dataset

ϵ	Time(hour)	Memory(MB)	#Cand.	#Prun.
HuspExt				
0.00004	1.882	987	0.752	196.395
0.00006	1.868	960	0.304	140.015
0.00008	1.847	958	0.162	111.248
0.00010	1.829	958	0.107	93.613
0.00012	1.827	957	0.077	81.536
SWU				
0.00004	No Resp. in 2.5h			
0.00006	No Resp. in 2.5h			
0.00008	2.394	973	0.536	128.649
0.00010	2.349	971	0.271	107.410
0.00012	2.290	970	0.175	92.975

The results of the evaluation with the Chain-Store dataset is provided in Table 6.6. In this table, last two columns display the number of generated and pruned candidates respectively in millions. We have made experiments with HuspExt and SWU. SWU cannot respond in 2.5 hours for threshold values 0.00004 and 0.00006. HuspExt outperforms SWU for the threshold values 0.00008-0.00012. The time and memory consumption together with the number of generated candidates and the pruned nodes increase as the minimum utility threshold decreases, which is an expected result. However, the important point to emphasize here is that, the proposed solution can respond even for very small thresholds such as 0.00004 without running out of

memory for such a sparse dataset. The memory consumption and execution time is more than other datasets, however, it is still feasible under the test environment and the nature of the data.

6.2 Evaluation of the Proposed Solution for Sequential Pattern Extraction Under User-Defined Pattern Scoring

In this section, details of the evaluation of our second solution to sequential pattern extraction, which uses user-defined pattern scoring in order to extract patterns, is presented. For the experiments, an Intel Core i5, 2.67 GHz computer with a 4GB main memory running Microsoft Windows 7 is used and the programs are coded in Java, in Oracle JDeveloper framework.

Table 6.7: Dataset characteristics

Dataset	#Accesses	#Sessions	#Distinct Requests	#Users
NASA	737148	90707	2459	6406
SASKA	288727	40784	2285	16664
MSNBC	264120	46020	17	17824
CTI	14490	13745	683	485
CENG	1752771	304567	210026	9671

6.2.1 DataSets

In the experiments, we use five different datasets, namely, *NASA, SASKA, MSNBC, CTI* and *CENG* datasets, whose basic specifications are given in Table 6.7.

The *NASA* dataset is from the NASA Kennedy Space Center server from July 1995 to August 1995[3]. It originally contains 3,461,612 page requests. After the data preparation phase, we have 737,148 accesses, 90,707 sessions, 2459 distinct requests, and 6406 distinct users.

[3] http://ita.ee.lbl.gov/html/contrib/NASA-HTTP.html

The second dataset, *SASKA*, is from the University of Saskatchewan's WWW server[4]. This server log was collected from June 1995 through to December 1995. The dataset originally contains 2,408,625 requests. After preprocessing, 288727 accesses, 40784 sessions, 2285 distinct requests, and 16664 distinct users are left.

The *MSNBC* dataset[5] includes the page visits of users who visited msnbc.com on 28.09.1999. The visits are recorded at the level of URL category (for example sports, news, etc.) and it includes visits to 17 categories. Therefore, 17 distinct accesses exists. Number of users in the dataset after preprocessing is 17824 and the number of URLs per category changes between 10 and 5000.

The *CTI* dataset[6] includes the sessionized data for the main DePaul CTI web server based on a random sample of users visiting the site for a two-week period during April 2002. The dataset includes 683 distinct accesses and 13745 distinct user sessions.

The *CENG* dataset is from the Computer Engineering (CENG) Department of Middle East Technical University (METU)[7]. The dataset consists of many sub-web sites including web pages of individuals (i.e., students, teachers), newsgroups, and courses. The web server logs are from 03.07.2011 to 13.11.2011, and the dataset originally contains 7,041,032 accesses. After preprocessing, there remains 1,752,771 accesses and 304,567 distinct sessions.

Datasets used in the evaluation are selected based on the fact that, they have different characteristics in terms of web site context and number of pageviews. In addition, NASA, SASKA, MSNBC, and CTI datasets are well-known public datasets. Specifically, both NASA and MSNBC datasets include visits to big portals, which essentially translate to a high number of sessions with very long paths, whereas the CTI, SASKA and CENG datasets refers to academic web sites. MSNBC dataset, on the other hand, has the characteristic of very few pageviews, since the visits are recorded at the level of web page categories.

[4] http://ita.ee.lbl.gov/html/contrib/Sask-HTTP.html
[5] http://kdd.ics.uci.edu/databases/msnbc/msnbc.html
[6] http://maya.cs.depaul.edu/ classes/ect584/data/cti-data.zip
[7] http://www.ceng.metu.edu.tr

6.2.2 Evaluation Methodology and Metrics

To evaluate the presented solution, a case study has been designed so as to clearly demonstrate the performance of the system under a real-world scenario and a possible application of the technique to an existing problem. In the case study, extracted patterns are considered as *paths* that can be recommended to users so as to assist their navigation in the web sites. For the experiments, we perform 5-fold cross-validation for each dataset. In each of the five iterations, each dataset is divided into training (70%) and evaluation (30%) datasets. We form the clusters from the training set, and we extract paths for each cluster. Evaluation of the extracted patterns for each test session is performed in four steps:

1. assigning test session to a cluster,
2. finding the patterns that match the test session,
3. forming a list of all matching paths,
4. checking whether the paths in the list are successful.

The first step assigns each test session to a cluster by comparing the session and the *cluster means*. A test session is composed of two parts. The first *70%* of the accesses is used to match a path and the rest *30%* is used for finding out whether the path is successful. In the second step, each path in the assigned cluster is checked to find out whether the first *70%* of the accesses in the test session contains the *path except for its last access*. If it is the case, that path is added to the list. We consider a path in the list to be successful if its last access exists in 30% of the second part of the test session. Metrics are calculated considering this evaluation process.

For measuring the performance of the system, we used *coverage and accuracy* metrics. The following parameters are used in the formulation of them:

- p_exc_l: path except its last access
- $session\_count$: total number of sessions
- $session\_count\_p$: number of sessions that any of p_exc_l exists

- $session\_count\_p\_succ$: number of sessions that any of the paths becomes successful

Coverage concerns the degree to which the extracted patterns cover the set of test sessions. To achieve this, the number of the test sessions that any of the path (except its last access) exists in, is found. Coverage is calculated as shown in Equation 6.1.

$$Coverage = session\_count\_p/session\_count \tag{6.1}$$

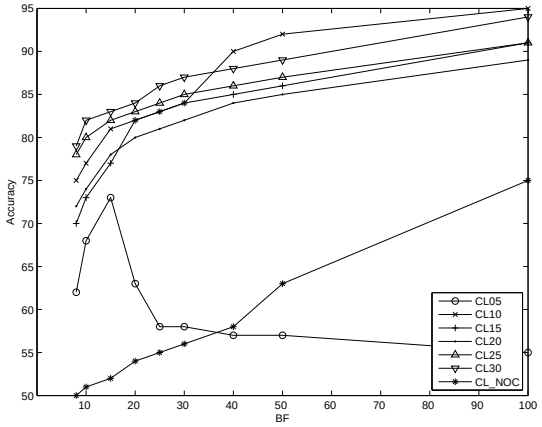
Accuracy measures the degree to which the extracted patterns successfully summarize the data. Considering our system, we look at each test session and find out whether any of the path except its last access (sp_exc_l) exist in that session. If exists, the rest of the session is examined to find whether the last access of our path exists in the rest of the session. Accuracy is calculated as shown in Equation 6.2.

$$Accuracy = session\_count\_p\_succ/session\_count\_p \tag{6.2}$$

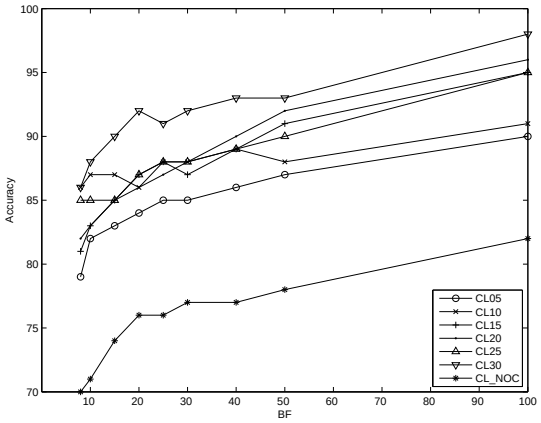
In addition to the above metrics, we calculated the average number of patterns extracted for a test session and reported in the results.

6.2.3 Results

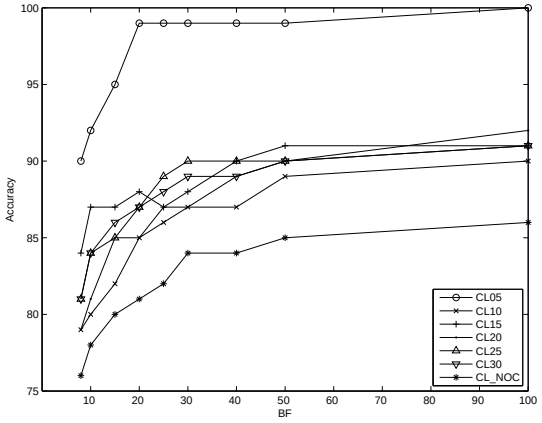
For the evaluation of the proposed technique, we have made experiments on the solution parameters so as to find out the effect of changing the size of the search space, number of clusters and weight parameters on the efficiency of the solution under datasets of different volumes and domains. In addition, in order to test the value difference between the patterns identified by our solution and that the patterns identified purely by frequent sequential pattern mining, we have made experiments with PrefixSpan algorithm. For all of the experiments, parameter values are chosen in an unbiased manner such that, the proposed algorithm can respond within the computational limits of the evaluation environment.



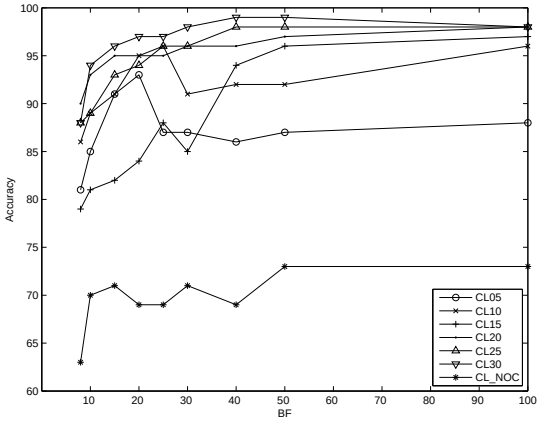
(a) NASA



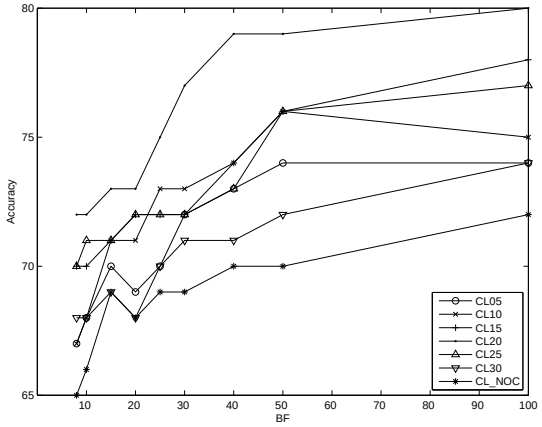
(b) SASKA



(c) MSNBC

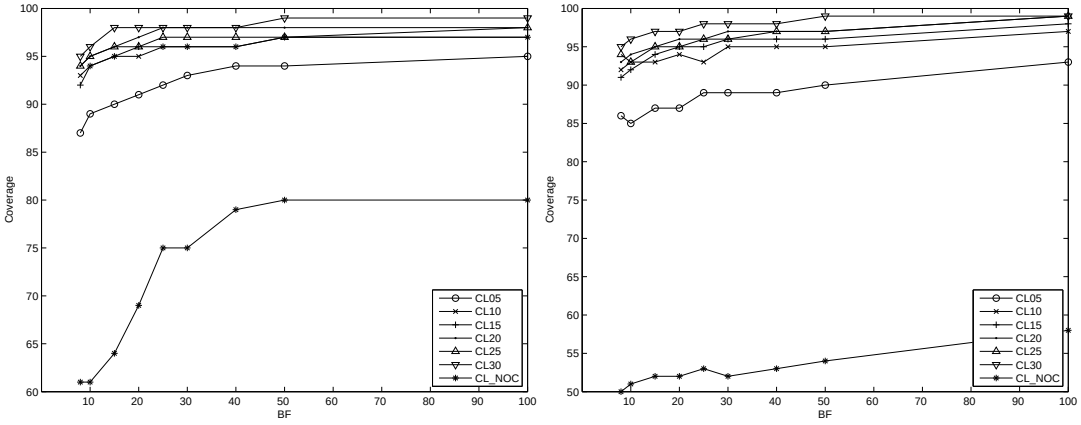


(d) CTI

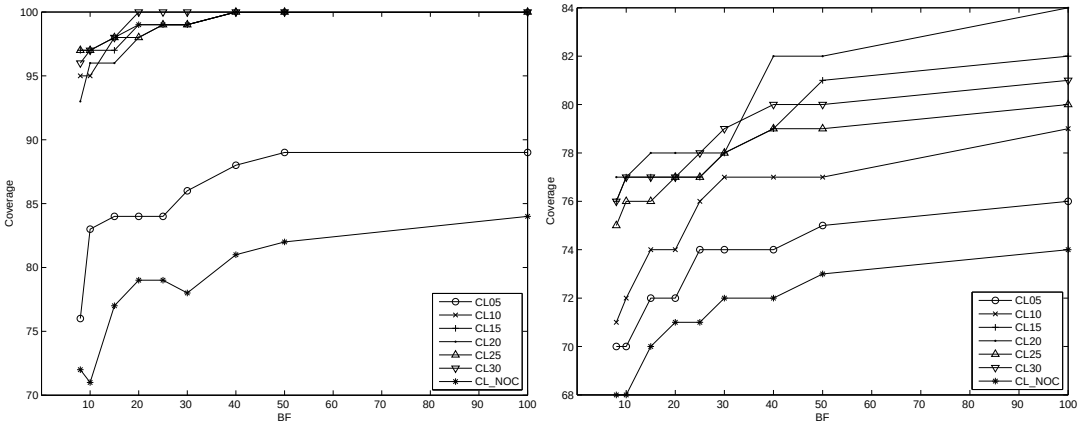


(e) CENG

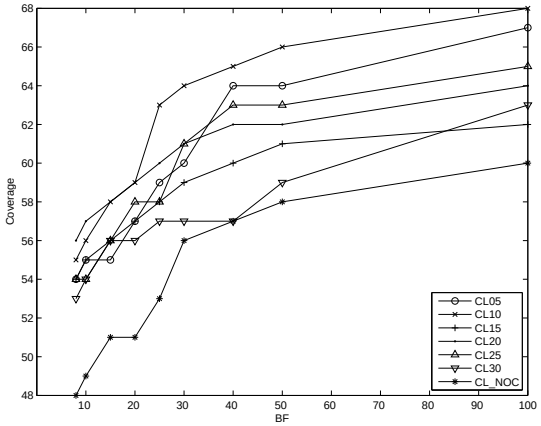
Figure 6.6: Accuracy results for different datasets



(a) NASA (b) SASKA



(c) MSNBC (d) CTI



(e) CENG

Figure 6.7: Coverage results for different datasets

Table 6.8: Average number of extracted patterns versus BF

NASA							
BF	NO_CL	CL05	CL10	CL15	CL20	CL25	CL30
8	1.16	3.25	4.61	4.68	5.43	6.85	5.94
10	1.71	3.95	5.59	5.54	5.96	7.33	6.76
15	1.88	3.75	6.78	7.46	7.53	7.89	8.37
20	2.10	4.25	8.34	9.23	7.39	9.44	8.94
25	2.10	4.89	7.92	7.83	8.06	11.23	9.43
30	2.59	5.30	8.28	8.33	8.61	1.26	9.83
40	2.94	6.92	8.72	9.12	7.81	12.01	10.58
50	3.39	7.42	9.84	10.05	9.05	12.21	11.46
100	4.78	9.78	11.03	10.30	10.04	12.79	11.90
CENG							
BF	NO_CL	CL05	CL10	CL15	CL20	CL25	CL30
8	0.51	0.60	0.69	0.71	0.92	0.93	1.23
10	0.54	0.61	0.79	0.82	1.00	1.19	1.49
15	0.57	0.69	0.96	1.07	1.35	1.54	1.87
20	0.63	0.75	1.18	1.28	1.67	1.92	2.35
25	0.78	0.88	1.59	1.71	2.05	2.29	2.69
30	0.98	1.14	1.72	1.92	2.53	3.87	2.74
40	1.41	1.42	1.97	2.32	3.08	4.89	3.47
50	1.59	1.69 6	2.20	4.22	3.32	7.88	4.78
100	2.35	2.01	2.64	5.61	4.65	9.61	5.91

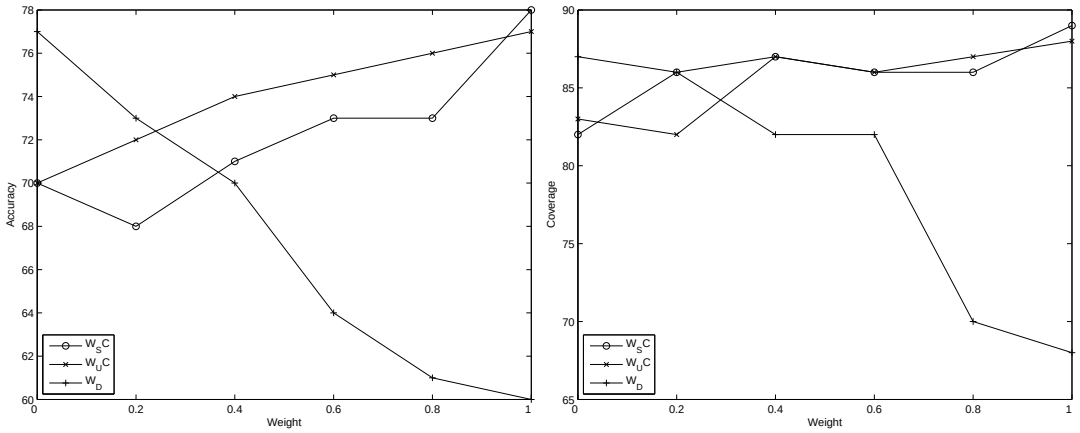
6.2.3.1 Experiments on BF and the Cluster Count

We perform tests with different number of clusters ranging from *5* to *30* and *BF* for values *{8, 10, 15, 20, 25, 30, 40, 50, 100}*. In order to find the effect of clustering, we conduct experiments when clustering is not performed. The results of these experiments are obtained by fixing $w\_session\_count$, w_user_count, and $w\_duration$ parameters to *0.33, 0.33 and 0.34*, respectively.

Figures 6.6 and 6.7 plot the *accuracy* and *coverage* results, respectively. In these figures, $CL05$ corresponds to the configuration of the system where *5* clusters is used, and the others are labeled in the same way.

For all of the datasets, when no clustering is performed, the performance of the system is lower than all other configurations independent of the number of clusters used. This result is valuable for us to see how clustering increases both the *coverage* and the *accuracy* of the solution. Clustering enables the system to extract more patterns where these patterns better reveal the users' interests. In addition, when we examine the effect of the number of clusters on the accuracy of the system, for the NASA dataset, for *BF* smaller than or equal to *15*, accuracy increases as the number of clusters increases for more than *80%* of the cases. For the SASKA dataset, the behavior of the accuracy is similar. For *30* clusters, the system always receives best performance for all *BF* values. Similarly, for the CTI dataset, highest accuracy results are achieved with *30* clusters for all *BF* values greater than *8*. For the MSNBC dataset, clustering the data to *5* groups gives best performance for all *BF* values. Finally with *20* clusters, CENG dataset achieved best accuracy for all *BF* values.

From these results, we observe that the proper number of clusters to use in the system depends on the nature of the dataset. Large websites with many possible requests results in identifying *many different usage behaviors* by clustering. NASA and SASKA are examples of such datasets. However, MSNBC has a smaller number of possible navigation paths in comparison to NASA and SASKA, which results in less number of usage characteristic groups. In addition,when *coverage* results are examined, it is observed that *coverage* increases in parallel with the *accuracy* for different cluster counts. Considering these results, we can conclude that, clustering effects the perfor-



(a) NASA (b) CENG

Figure 6.8: Accuracy versus weight parameters for different datasets

mance of the system and its affect depends on the nature of the web site in terms of distinct usage characteristics. If we cluster the dataset such that it is not partitioned adequately to reflect coherent groups, the performance decreases. As the number of distinct usage groups of a web site increases, the number of clusters should increase in order to better reflect user's interests.

When we examine the effect of *BF* on the *accuracy* and *coverage*, we observe that, in more than 80% of the cases, both *accuracy* and *coverage* increase as the number of child nodes kept at each level increases. This results from the fact that, as *BF* increases, more patterns can be constructed. In other words, the algorithm gives more chance to patterns that have lower evaluation values than its siblings at the same level, but have higher evaluation values than its siblings at lower levels. However, one other effect of the increased *coverage* is the increased number of estimations, which decreases the *accuracy* at some of the cases.

In order to have an idea about the number of patterns the system extracts, we have displayed the average number of extracted patterns for the NASA and CENG datasets in Table 6.8. Considering the same parameter settings for *BF* and *number of clusters*, for the NASA dataset, which is a comprehensive web portal, the average number of extracted patterns is greater than the CENG dataset, which is rather a smaller web site. In addition, as *BF* increases, *number of extracted patterns* also increases. This confirms with the previous discussions. As more child nodes are kept during WaPUPS,

more patterns are extracted.

6.2.3.2 Experiments on the Evaluation Function Parameters

In order to find the effect of evaluation function parameters, namely, $w\_session\_count$ (W_SC), w_user_count (W_UC) and $w\_duration$ (W_D), we conducted experiments using WaPUPS for all of the five datasets. We included only the results for NASA and CENG datasets, since for the other datasets, results are similar and the conclusions we reach are the same when the effect of the weight parameters on the system's accuracy is concerned.

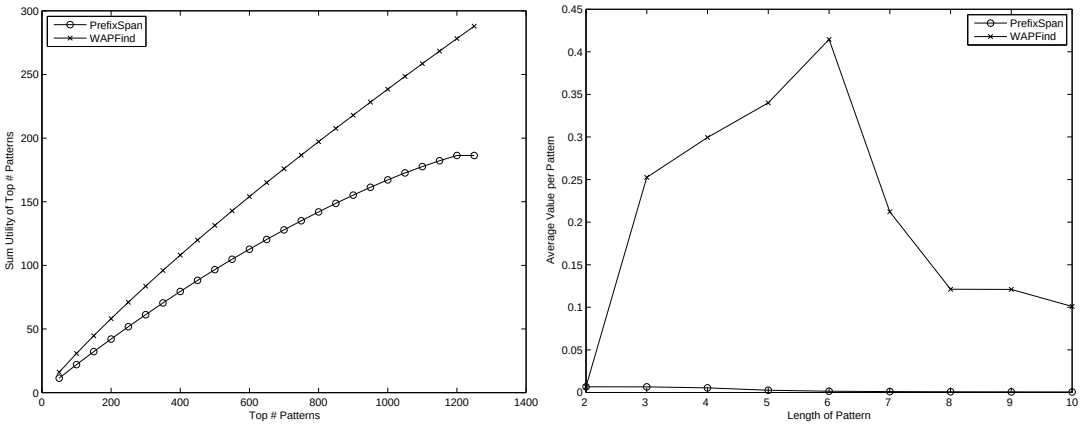
The parameters are tested for the values *{0, 0.2, 0.4, 0.6, 0.8, 1}*. The sum of these parameters are always *1* in the system. Results of these experiments are obtained by fixing the *number of clusters* and *BF* to *10*. Figure 6.8 displays the *accuracy* of the proposed approach as we change the weight parameters for the NASA and the CENG datasets.

For both of the datasets, the behavior of the weight parameters is similar. As $w\_session\_count$ and w_user_count increase, *accuracy* increases. However, as the weight of the duration increases, *accuracy* decreases. In addition, covering more distinct user sessions increases the performance of the solution. The results, however, show an inverse relationship between the *accuracy* of the system and $w\_duration$. Considering these, we can conclude that the best setting for the parameters should assign higher values to $w\_session\_count$ and w_user_count in comparison with $w\_duration$. However, since the solution is based on user defined evaluation criteria, we believe that, the construction of the evaluation function itself, and therefore, the best setting for the weight parameters should be performed in cooperation with the pattern users.

6.2.3.3 Comparison with Frequent Pattern Mining

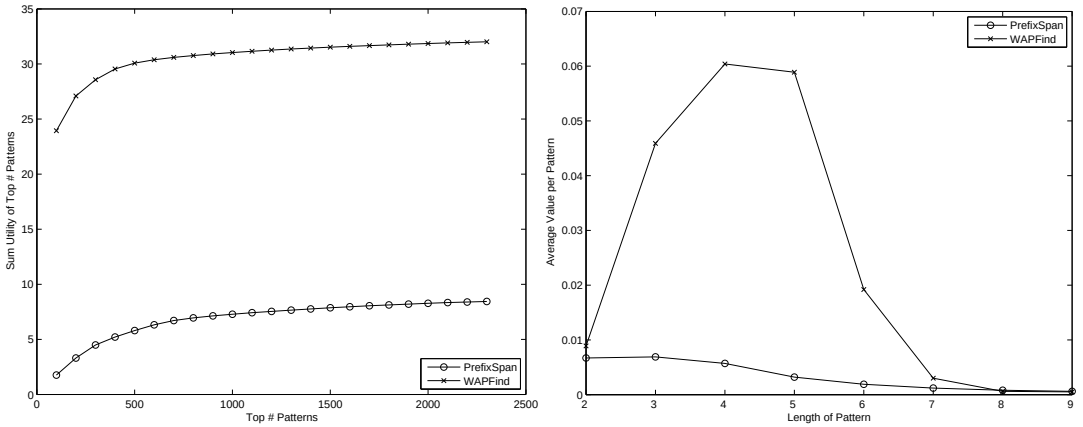
We made experiments with PrefixSpan algorithm [8] so as to compare our solution with frequent sequential pattern mining. In order to achieve this, we have made two sets

[8] We used Prefixspan implementation given in http://www.philippe-fournier-viger.com/spmf/



(a) (b)

Figure 6.9: Comparison of value results of WaPUPS and PrefixSpan for the NASA dataset



(a) (b)

Figure 6.10: Comparison of value results of WaPUPS and PrefixSpan for the CENG dataset

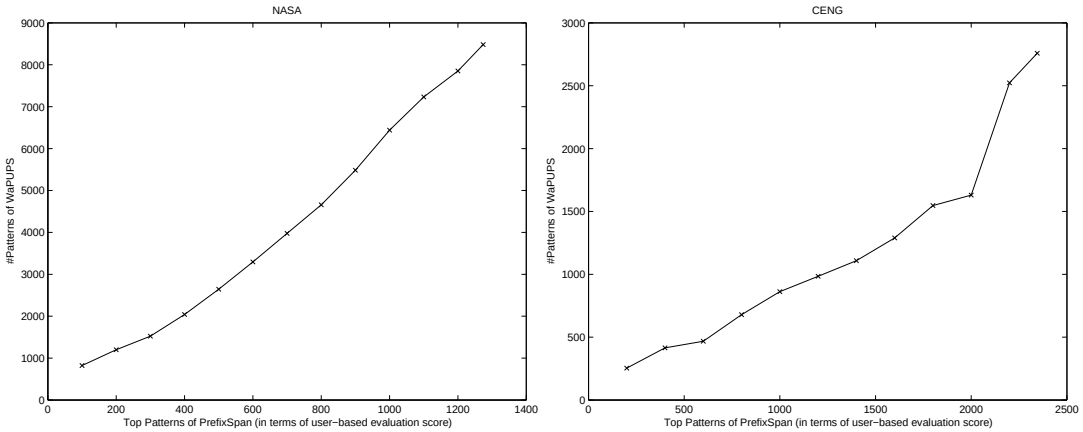
of experiments. In the first set, the values of the patterns extracted by WaPUPS and by frequent sequential pattern mining are compared. In the second set, we aim to measure the performance of our method as to what is the hit ratio of high valued patterns.

Results of the first set of experiments where the extracted pattern scores of WaPUPS and PrefixSpan compared are displayed in Figures 6.9 and 6.10 for NASA and CENG datasets, respectively. In these figures, for the graphs in the first column, x-axis refers to the top n number of patterns extracted by PrefixSpan and WaPUPS. On the other hand, y-axis represents the sum of the scores of the top n extracted patterns. In addition, the graphs in the second column display the average values of the extracted patterns with respect to different pattern lengths, where x-axis refers to the lengths of patterns and y-axis shows the average values per pattern. When these figures are examined, it can be concluded that the proposed solution can identify higher valued patterns when both cumulated sum of the top valued patterns and average value of the extracted patterns with respect to different pattern lengths are concerned. Although, the amount of pattern scores accumulated by the proposed algorithm is always higher than that of PrefixSpan for both the sum and average cases, the rate of change in the pattern scores is not same for NASA and CENG datasets. The behaviour is not steady, but on the overall there is an increase and the rate of change may vary depending on the dataset.

There exists two motivations behind the second set of experiments. First of all, our proposed algorithm is not a complete algorithm since it is based on the heuristic that the best BF patterns at each level leads to the extraction of high valued patterns in terms of user's evaluation scoring. Therefore, we aim to find out the hit ratio of high valued patterns considering all possible sequential patterns that can be extracted from the datasets under consideration. Second, although BF results in an algorithm that is not complete, it allows managing the size of the search space and extract highest valued patterns among all the extracted patterns which is a solution to one of the main challenges of sequential pattern mining research.

In this experiment, we aim to extract all exiting patterns, i.e., sequential patterns having non-zero support. However, practically this is not feasible. Instead, we low-

ered support threshold as much as possible and aimed to extract all the sequence patterns we could. We found user's evaluation score for each pattern and consider this set in the experiment. We applied this procedure to all datasets; however, we cannot take response from PrefixSpan for the datasets under consideration, since it cannot complete processing for low support values due to memory limitations of the experimental environment. For the support values that PrefixSpan can complete its processing, it cannot generate any patterns. One important observation to point out here is that, for those datasets, WaPUPS can extract patterns by controlling the size of the search space using the BF parameter. In addition, it extracts patterns with highest scores among all the extracted patterns. Therefore, for this experiment, we used only the *73.7%* and *78.2%* of the NASA and CENG datasets where we eliminated sessions that has accesses greater than 40 and 25, respectively. We used *support values 0.015 for NASA and 0.007 for CENG dataset. For WaPUPS,* number of clusters, BF, $w\_session\_count$, w_user_count and $w\_duration$ parameters are set to *20, 100, 0.33, 0.33* and *0.34*, respectively.



(a) (b)

Figure 6.11: Comparison of number of patterns extracted by WaPUPS and PrefixSpan

In this second set of experiments, we compare the top *k* patterns extracted by WaPUPS and PrefixSpan in terms of user based evaluation score. In Figure 6.11 we present the hit ratio in top *k* patterns. For the graphs in this figure, we divide the set of patterns extracted by PrefixSpan into value intervals represented in x-axis. Y-axis displays the total number of patterns extracted by WaPUPS that fall into these value intervals. First of all, extracted patterns of WaPUPS almost uniformly fall into the value intervals

defined by the top valued patterns of PrefixSpan. Totally, WaPUPS extracted *8503* and *2880* patterns from NASA and CENG datasets, respectively. In addition *274* and *129* of the patterns extracted by WaPUPS have scores greater than the highest valued pattern of PrefixSpan for NASA and CENG datasets, respectively. The ratio of the value of the maximum valued pattern of WAPUPS to that of the PrefixSpan are *1.348* for NASA and *13.402* for CENG dataset. When all these results are considered, it can be concluded that, patterns extracted by WaPUPS have significantly higher evaluation values than the patterns extracted by PrefixSpan. In addition, with BF, solution can always generate patterns by controlling the size of the search space. Overall, it can be concluded that, the proposed solution can generate better patterns in terms of the evaluation criteria set by the users.

CHAPTER 7

CONCLUSION

Sequential pattern mining is one of the fundamental research areas of data mining that has a wide range of application areas including bioinformatics, customer behaviour analysis and marketing applications. Sequence data on its own is generally big and for some domains very sparse, therefore, the efficiency of the proposed solutions are extremely important. More specifically, how effectively the solutions prune the search space and therefore lower the time and memory requirements are important considerations. In addition to this, most of the solutions proposed for sequential pattern extraction based their value analysis of the patterns on the support measure. However, in many application domains, frequency is not the only desired property. Even in some scenarious, users may be interested in rare patterns. An example to such a real life scenario can be given from the medical domain. For instance, doctors may be interested in extracting patterns that are rare in the data which will give them clue about rare symptoms. Therefore, the main challenges and limitations of sequential pattern mining are related to; first the efficiency of the proposed approaches, and second, the limitation of the existing frequency based techniques.

In this thesis work, we propose two solutions to sequential pattern extraction. The first solution is a new framework in the area of utility-based sequential pattern mining, which is one of the solutions proposed by the researchers in order to extract patterns based on the utility instead of frequency. In utility-based sequential pattern mining, a new value definition for the patterns is introduced, which is called utility. In this research problem, the aim is to extract patterns that have utility value higher than or equal to a user defined minimum utility threshold. The proposed solution in this area

is a generic framework for high utility sequential pattern extraction and the solution defines and formalizes CRoM, which is a tighter upper bound on the utility of the candidate patterns in comparison with the state of the art TWU-based upper bound. As proved formally, CRoM correctly eliminates candidate items. The proposed ideas are realized by HuspExt algorithm, in which CRoM is utilized for filtering out of the unpromising items prior to candidate generation. Substantial experiments on both synthetic and real datasets have shown that, HuspExt can extract a complete set of patterns in lower time and memory requirements even in largescale data with the utilization of low utility threshold values. By adopting a tighter upper bound and efficient data structures to realize the pruning method, we can decrease the number of candidate items and enhance the performance of the utility based sequential pattern mining framework.

Although utility-based sequential pattern mining presents a solution for the cases where the value of a pattern is not limited to its frequency, utility framework is not always adequate for defining the value of a pattern. More specifically, in some real life cases, the value of a pattern does not depend on distinct item utilities as it is defined in classical utility-based framework. For example, the demographic information about the users of the patterns can define the value of the extracted patterns. In addition, common to both frequency-based and utility-based techniques, one important challenge is that, there exists a huge number of possible patterns that are hidden in databases and the mining algorithm should find the set of patterns, when possible, satisfying the minimum support or the minimum utility threshold. For especially huge and sparse datasets, however, it is not possible to extract patterns under high threshold values. On the other hand, even state of the art algorithms may fail to respond under even very low threshold values.

In order to handle these limitations, in this thesis work, a new solution is defined for extracting sequential patterns based on user defined pattern scoring. The solution defines an evaluation function which determines the value of the patterns. This function can be defined based on any factors that are important for the end-users considering the value of the extracted patterns. In addition, we control the size of the search space by defining a new parameter, *BF (Branching Factor)*, which enables the solution to extract high-valued patterns even in the cases for which existing frequency-based

techniques and utility-based techniques cannot generate any pattern.

The current version of the solution is evaluated in the web usage domain, therefore it is adapted to work with web server log files. However, it should be mentioned that, the proposed technique can work with any other sequence data, as well. The solution is based on a novel approach that combines clustering with a new pattern extraction algorithm. The role of clustering in the framework is to group user sessions on the basis of similar navigational behaviours that increase the accuracy of the system as shown by the experiments. For the pattern extraction phase, the core of the proposed algorithm is evaluation function which captures the user's definition for the value of a pattern. We propose WaPUPS algorithm to traverse the search space for high-valued patterns. In addition, in order to examine how the systems performance changes when more patterns are allowed to grow, we implemented and tested another version of WaPUPS. In that version, child nodes of each parent are considered separately and best BF children of each parent are allowed to proceed to the next level. This increases the search space at each level from BF to BF times the number of nodes that exist in the previous level. However, from the results of the experiments we made with that version, we examine that, this enhancement does not improve the performance of the solution; therefore it is not included in the thesis work.

Experimental results taken from this solution have shown that proposed approach is capable of effectively discovering user access patterns and revealing the underlying value defined by the evaluation function. In addition, using the BF parameter to control the size of the search space, it can generate high scored patterns even for the cases where the state of art techniques cannot extract any pattern.

Future work for the thesis study will focus on both of the proposed solutions. First of all, for the first solution, our plan is to adapt the solution to work with cases where the utility of the patterns can be negative. In addition, we will work on performance issues in terms of both time and space. We are planning to work on different techniques so as to lower both the time and memory requirements of the proposed solution.

For the second solution, the future work will focus on adapting the system to work on other domains as well. Therefore, we are planning to evaluate the proposed solution under datasets from different domains and perform experiments over sequence

datasets other than the web log data, as well.

As another future direction, the ideas from the two proposed solutions can be integrated. One possible way of integration is to apply the BF parameter to utility based sequence mining so as to improve efficiency and to be able to respond under low utility thresholds. In addition, it is possible to investigate the integration of both of the solutions with conventional frequency based filtering in order to extract the high frequency patterns having high utility.

REFERENCES

[1] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules in large databases. In *Proceedings of the 20th International Conference on Very Large Data Bases*, VLDB '94, pages 487–499, San Francisco, CA, USA, 1994. Morgan Kaufmann Publishers Inc.

[2] Rakesh Agrawal and Ramakrishnan Srikant. Mining sequential patterns. In *Proceedings of the Eleventh International Conference on Data Engineering*, ICDE '95, pages 3–14, Washington, DC, USA, 1995. IEEE Computer Society.

[3] Chowdhury Farhan Ahmed, Syed Khairuzzaman Tanbeer, and Byeong-Soo Jeong. Mining high utility web access sequences in dynamic web log data. In *Proceedings of the 2010 11th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing*, SNPD '10, pages 76–81, Washington, DC, USA, 2010. IEEE Computer Society.

[4] Chowdhury Farhan Ahmed, Syed Khairuzzaman Tanbeer, and Byeong-Soo Jeong. A novel approach for mining high-utility sequential patterns in sequence databases. *2010 ETRI Journal*, 32:676–686, 2010.

[5] Chowdhury Farhan Ahmed, Syed Khairuzzaman Tanbeer, Byeong-Soo Jeong, and Young-Koo Lee. Efficient mining of utility-based web path traversal patterns. In *Advanced Communication Technology, 2009. ICACT 2009. 11th International Conference on*, volume 03, pages 2215–2218, Feb 2009.

[6] Chowdhury Farhan Ahmed, Syed Khairuzzaman Tanbeer, Byeong-Soo Jeong, and Young-Koo Lee. Efficient tree structures for high utility pattern mining in incremental databases. *IEEE Trans. Knowl. Data Eng.*, 21(12):1708–1721, 2009.

[7] Oznur Kirmemis Alkan and Pinar Senkul. Intweb: An ai-based approach for adaptive web. In *Proceedings of the 9th Workshop on Intelligent Techniques for Web Personalization*, 2011.

[8] Oznur Kirmemis Alkan and Pinar Senkul. Assisting web site navigation through web usage patterns. In *Recent Trends in Applied Artificial Intelligence, 26th International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems, IEA/AIE 2013, Amsterdam, The Netherlands, June 17-21, 2013. Proceedings*, pages 161–170, 2013.

[9] Oznur Kirmemis Alkan and Pinar Senkul. Extracting sequential patterns based on user defined criteria. In *Hybrid Artificial Intelligent Systems - 8th International Conference, HAIS 2013, Salamanca, Spain, September 11-13, 2013. Proceedings*, pages 181–190, 2013.

[10] Oznur Kirmemis Alkan and Pinar Senkul. New techniques for adapting web site topology and ontology to user behavior. In Erol Gelenbe and Ricardo Lent, editors, *Computer and Information Sciences III*, pages 419–427. Springer London, 2013.

[11] Oznur Kirmemis Alkan and Pinar Senkul. Crom and huspext: Improving efficiency of high utility sequential pattern extraction. *Knowledge and Data Engineering, IEEE Transactions on*, PP(99):1–1, 2015.

[12] Jay Ayres, Jason Flannick, Johannes Gehrke, and Tomi Yiu. Sequential pattern mining using a bitmap representation. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '02, pages 429–435, New York, NY, USA, 2002. ACM.

[13] Arindam Banerjee and Joydeep Ghosh. Clickstream clustering using weighted longest common subsequences. In *In Proceedings of the Web Mining Workshop at the 1st SIAM Conference on Data Mining*, pages 33–40, 2001.

[14] Mostafa Haghir Chehreghani. Efficiently mining unordered trees. In *Data Mining (ICDM), 2011 IEEE 11th International Conference on*, pages 111–120, Dec 2011.

[15] Kiran Chelluri and Vijay Kumar. Data classification and management in very large data warehouses. In *Third International Workshop on Advanced Issues of E-Commerce and Web-Based Information Systems (WECWIS '01), San Jose, California, USA, June 21-22, 2001*, pages 52–57, 2001.

[16] Enhong Chen, Huanhuan Cao, Qing Li, and Tieyun Qian. Efficient strategies for tough aggregate constraint-based sequential pattern mining. *Information Sciences*, 178(6):1498 – 1518, 2008.

[17] Robert Cooley, Bamshad Mobasher, and Jaideep Srivastava. Web mining: Information and pattern discovery on the world wide web. In *ICTAI '97: Proceedings of the 9th International Conference on Tools with Artificial Intelligence*, page 558, Washington, DC, USA, 1997. IEEE Computer Society.

[18] Robert Cooley, Bamshad Mobasher, and Jaideep Srivastava. Data preparation for mining world wide web browsing patterns. *Knowledge and Information Systems*, 1(1):5–32, 1999.

[19] Mariam Daoud, Lynda-Tamine Lechani, and Mohand Boughanem. Towards a graph-based user profile modeling for a session-based personalized search. *Knowledge and Information Systems*, 21(3):365–398, 2009.

[20] Nadav Eiron and Kevin S. McCurley. Untangling compound documents on the web. In *Proceedings of the Fourteenth ACM Conference on Hypertext and Hypermedia*, HYPERTEXT '03, pages 85–94, New York, NY, USA, 2003. ACM.

[21] Oren Etzioni. The world-wide web: Quagmire or gold mine? *Commun. ACM*, 39(11):65–68, November 1996.

[22] Usama M. Fayyad, Gregory Piatetsky-Shapiro, and Padhraic Smyth. Advances in knowledge discovery and data mining. chapter From Data Mining to Knowledge Discovery: An Overview, pages 1–34. American Association for Artificial Intelligence, Menlo Park, CA, USA, 1996.

[23] Gary William Flake, Steve Lawrence, C.Lee Giles, and Frans.M. Coetzee. Self-organization and identification of web communities. *Computer*, 35(3):66–70, Mar 2002.

[24] Yongjian Fu, Kanwalpreet Sandhu, and Ming-Yi Shih. Clustering of web users based on access patterns. In *In Proceedings of the 1999 KDD Workshop on Web Mining*. Springer-Verlag, 1999.

[25] Minos N. Garofalakis, Rajeev Rastogi, and Kyuseok Shim. Spirit: Sequential pattern mining with regular expression constraints. In *Proceedings of the 25th International Conference on Very Large Data Bases*, VLDB '99, pages 223–234, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.

[26] Mathias Géry and Hatem Haddad. Evaluation of web usage mining approaches for user's next request prediction. In *Proceedings of the 5th ACM International Workshop on Web Information and Data Management*, WIDM '03, pages 74–81, New York, NY, USA, 2003. ACM.

[27] Abdelghani Guerbas, Omar Addam, Omar Zarour, Mohamad Nagi, Ahmad Elhajj, Mick J. Ridley, and Reda Alhajj. Effective web log mining and online navigational pattern prediction. *Knowl.-Based Syst.*, 49:50–62, 2013.

[28] J. Han, J. Pei, and X. Yan. Sequential pattern mining by pattern-growth: Principles and extensions*. In Wesley Chu and Tsau Young Lin, editors, *Foundations and Advances in Data Mining*, volume 180 of *Studies in Fuzziness and Soft Computing*, pages 183–220. Springer Berlin Heidelberg, 2005.

[29] Jiawei Han and Micheline Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2000.

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

[31] Jiawei Han, Jian Pei, and Yiwen Yin. Mining frequent patterns without candidate generation. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, SIGMOD '00, pages 1–12, New York, NY, USA, 2000. ACM.

[32] Birgit Hay, Geert Wets, and Koen Vanhoof. Mining navigation patterns using a sequence alignment method. *Knowledge and Information Systems*, 6(2):150–163, 2004.

[33] Yin-Fu Huang and Jhao-Min Hsu. Mining web logs to improve hit ratios of prefetching and caching. *Know.-Based Syst.*, 21(1):62–69, February 2008.

[34] Guo-Cheng Lan, Tzung-Pei Hong, Vincent S. Tseng, and Shyue-Liang Wang. Applying the maximum utility measure in high utility sequential pattern mining. *Expert Systems with Applications*, 41(11):5071 – 5081, 2014.

[35] Yuefeng Li and Ning Zhong. Web mining model and its applications for information gathering. *Knowledge-Based Systems*, 17:207 – 217, 2004. Special Issue: Web Intelligence.

[36] Ying Liu, Wei Keng Liao, and Alok N. Choudhary. A two-phase algorithm for fast discovery of high utility itemsets. In Tu Bao Ho, David Wai-Lok Cheung, and Huan Liu, editors, *PAKDD*, volume 3518 of *Lecture Notes in Computer Science*, pages 689–695. Springer, 2005.

[37] Ying Liu, Wei-keng Liao, and Alok Choudhary. A fast high utility itemsets mining algorithm. In *Proceedings of the 1st International Workshop on Utility-based Data Mining*, UBDM '05, pages 90–99, New York, NY, USA, 2005. ACM.

[38] Nizar R. Mabroukeh and C. I. Ezeife. A taxonomy of sequential pattern mining algorithms. *ACM Comput. Surv.*, 43(1):3:1–3:41, December 2010.

[39] Heikki Mannila, Hannu Toivonen, and A. Inkeri Verkamo. Discovery of frequent episodes in event sequences. *Data Mining and Knowledge Discovery*, 1(3):259–289, 1997.

[40] F. Masseglia, F. Cathala, and P. Poncelet. The psp approach for mining sequential patterns. In JanM. Zytkow and Mohamed Quafafou, editors, *Principles of Data Mining and Knowledge Discovery*, volume 1510 of *Lecture Notes in Computer Science*, pages 176–184. Springer Berlin Heidelberg, 1998.

[41] Florent Masseglia, Pascal Poncelet, and Maguelonne Teisseire. Efficient mining of sequential patterns with time constraints: Reducing the combinations. *Expert Systems with Applications*, 36(2, Part 2):2677 – 2690, 2009.

[42] Stephen G. Matthews, Mario A. Gongora, Adrian A. Hopgood, and Samad Ahmadi. Web usage mining with evolutionary extraction of temporal fuzzy association rules. *Knowledge-Based Systems*, 54(0):66 – 72, 2013.

[43] Bamshad Mobasher. Data mining for web personalization. In Peter Brusilovsky, Alfred Kobsa, and Wolfgang Nejdl, editors, *The Adaptive Web*, volume 4321 of *Lecture Notes in Computer Science*, pages 90–135. Springer Berlin Heidelberg, 2007.

[44] Bamshad Mobasher, Honghua Dai, Tao Luo, and Miki Nakagawa. Discovery and evaluation of aggregate usage profiles for web personalization. *Data Mining and Knowledge Discovery*, 6:61–82, 2002.

[45] Carl H. Mooney and John F. Roddick. Sequential pattern mining – approaches and algorithms. *ACM Comput. Surv.*, 45(2):19:1–19:39, March 2013.

[46] Jian Pei, Jiawei Han, Behzad Mortazavi-Asl, Helen Pinto, Qiming Chen, Umeshwar Dayal, and Meichun Hsu. Prefixspan: Mining sequential patterns by prefix-projected growth. In *Proceedings of the 17th International Conference on Data Engineering*, pages 215–224, Washington, DC, USA, 2001. IEEE Computer Society.

[47] Jian Pei, Jiawei Han, Behzad Mortazavi-Asl, Jianyong Wang, Helen Pinto, Qiming Chen, Umeshwar Dayal, and Mei-Chun Hsu. Mining sequential patterns by pattern-growth: The prefixspan approach. *IEEE Trans. on Knowl. and Data Eng.*, 16(11):1424–1440, November 2004.

[48] Jay Pisharath, Jing Liu, Berkin Ozisikyilmaz, Ramanathan Narayanan, Weikeng Liao, Alok Choudhary, and Gokhan Memik. Nu-minebench version 2.0 data set and technical report.

[49] Jia-Dong Ren, Yin-Bo Cheng, and Liang-Liang Yang. An algorithm for mining generalized sequential patterns. In *Machine Learning and Cybernetics, 2004. Proceedings of 2004 International Conference on*, volume 2, pages 1288–1292 vol.2, Aug 2004.

[50] Bai-En Shie, Hui-Fang Hsiao, Vincent S. Tseng, and Philip S. Yu. Mining high utility mobile sequential patterns in mobile commerce environments. In *Proceedings of the 16th International Conference on Database Systems for Advanced Applications - Volume Part I*, DASFAA'11, pages 224–238, Berlin, Heidelberg, 2011. Springer-Verlag.

[51] Hou Sizu and Zhang Xianfei. Alarms association rules based on sequential pattern mining algorithm. In *Fuzzy Systems and Knowledge Discovery, 2008. FSKD '08. Fifth International Conference on*, volume 2, pages 556–560, Oct 2008.

[52] Myra Spiliopoulou. Web usage mining for web site evaluation. *Commun. ACM*, 43(8):127–134, August 2000.

[53] Myra Spiliopoulou, Lukas C. Faulstich, and Karsten Winkler. A data miner analyzing the navigational behaviour of web users. In *In Proc. of the Workshop on Machine Learning in User Modelling of the ACAI99*, 1999.

[54] Ramakrishnan Srikant and Rakesh Agrawal. Mining sequential patterns: Generalizations and performance improvements. In *Proceedings of the 5th International Conference on Extending Database Technology: Advances in Database Technology*, EDBT '96, pages 3–17, London, UK, UK, 1996. Springer-Verlag.

[55] Vincent S. Tseng, Bai-En Shie, Cheng-Wei Wu, and Philip S. Yu. Efficient algorithms for mining high utility itemsets from transactional databases. *IEEE Transactions on Knowledge and Data Engineering*, 25(8):1772–1786, 2013.

[56] Vincent S. Tseng, Bai-En Shie, Cheng-Wei Wu, and Philip S. Yu. Efficient algorithms for mining high utility itemsets from transactional databases. *IEEE Trans. on Knowl. and Data Eng.*, 25(8):1772–1786, August 2013.

[57] Jason Tsong-Li Wang, Gung-Wei Chirn, Thomas G. Marr, Bruce Shapiro, Dennis Shasha, and Kaizhong Zhang. Combinatorial pattern discovery for scientific data: Some preliminary results. In *Proceedings of the 1994 ACM SIGMOD International Conference on Management of Data*, SIGMOD '94, pages 115–125, New York, NY, USA, 1994. ACM.

[58] Xifeng Yan, Jiawei Han, and Ramin Afshar. Clospan: Mining closed sequential patterns in large datasets. In *In SDM*, pages 166–177, 2003.

[59] Hong Yao, Howard J. Hamilton, and Cory J. Butz. A foundational approach to mining itemset utilities from databases. In Michael W. Berry, Umeshwar Dayal, Chandrika Kamath, and David B. Skillicorn, editors, *SDM*. SIAM, 2004.

[60] Hong Yao, Howard J. Hamilton, and Liqiang Geng. A unified framework for utility-based measures for mining itemsets. in proc. of acm sigkdd 2nd workshop on utility-based data mining. In *Second International Workshop on Utility-Based Data Mining*, 2006.

[61] Junfu Yin, Zhigang Zheng, and Longbing Cao. Uspan: An efficient algorithm for mining high utility sequential patterns. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '12, pages 660–668, New York, NY, USA, 2012. ACM.

[62] Junfu Yin, Zhigang Zheng, Longbing Cao, Yin Song, and Wei Wei. Efficiently mining top-k high utility sequential patterns. In *Data Mining (ICDM), 2013 IEEE 13th International Conference on*, pages 1259–1264, Dec 2013.

[63] Unil Yun. A new framework for detecting weighted sequential patterns in large sequence databases. *Knowledge-Based Systems*, 21(2):110 – 122, 2008.

[64] Mohammed J. Zaki. Efficient enumeration of frequent sequences. In *Proceedings of the Seventh International Conference on Information and Knowledge Management*, CIKM '98, pages 68–75, New York, NY, USA, 1998. ACM.

[65] Mohammed J. Zaki. Spade: An efficient algorithm for mining frequent sequences. *Mach. Learn.*, 42(1-2):31–60, January 2001.

[66] Baoyao Zhou, Siu Cheung Hui, and Kuiyu Chang. An intelligent recommender system using sequential web access patterns. In *Cybernetics and Intelligent Systems, 2004 IEEE Conference on*, volume 1, pages 393–398 vol.1, Dec 2004.

[67] Lin Zhou, Ying Liu, Jing Wang, and Yong Shi. Utility-based web path traversal pattern mining. In *Data Mining Workshops, 2007. ICDM Workshops 2007. Seventh IEEE International Conference on*, pages 373–380, Oct 2007.

[68] Jianhan Zhu, Jun Hong, and John G. Hughes. Using markov chains for link prediction in adaptive web sites. In *In Proc. of ACM SIGWEB Hypertext*, pages 60–73. Springer, 2002.

CURRICULUM VITAE

PERSONAL INFORMATION

Surname, Name: Alkan Kirmemis, Oznur
Nationality: Turkish (TC)
Date and Place of Birth: 10.04.1982, Samsun
Marital Status: Married
Phone: 0 535 2447485

EDUCATION

Degree	Institution	Year of Graduation
Ph.D.	Dept. of Computer Engineering, METU	2015
M.S.	Dept. of Computer Engineering, METU	2008
B.S.	Dept. of Computer Engineering, METU	2004
High School	Atatürk High School, Ankara	1999

PROFESSIONAL EXPERIENCE

Year	Place	Enrollment
2014-	Dept. of Computer Engineering, YBU	Instructor
2010-2014	Republic of Turkey Prime Ministry	Computer Engineer
2006-2010	Siemens EC, Ankara	Computer Engineer
2004-2006	KoçSistem, Ankara	Software Specialist

PUBLICATIONS

International Journal Publications

1. O. Kirmemis Alkan, P. Karagoz (Under Review), WaPUPS: Web Access Pattern Extraction Under User-Defined Pattern Scoring, Submitted to Journal of Information Science on 10.04.2015.

1. O. Kirmemis Alkan, P. Karagoz, CRoM and HuspExt: Improving Efficiency of High Utility Sequential Pattern Extraction, IEEE Transactions on Knowledge and Data Engineering, 2015 (10.1109/TKDE.2015.2420557).

International Conference Publications

1. O. Kirmemis Alkan, P. Karagoz, Extracting Sequential Patterns Based on User Defined Criteria, International Conference on Hybrid Artificial Intelligence Systems (HAIS), Salamanca, Spain, September 2013.

2. O. Kirmemis Alkan, P. Karagoz , Assisting Web Site Navigation Through Web Usage Patterns, IEA/AIE 2013, pp.161-170, Amsterdam, Holland, June 2013.

3. O. Kirmemis, P. Senkul, New Techniques for Adapting Web Site Topology and Ontology to User Behavior", ISCIS 2012, Paris, France, October 2012.

4. O. Kirmemis Alan, P. Senkul , IntWEB: An AI-Based Approach for Adaptive Web, IJCAI 9th Workshop on Intelligent Techniques for Web Personalization, July 2011.

5. O. Kirmemis, A.Birturk, A Content-Based User Model Generation and Optimization Approach for Movie Recommendation, In 6th Workshop on Intelligent Techniques for Web Personalization and Recommender Systems at the Twenty-Third AAAI Conference on Artificial Intelligence (AAAI-08), 13-17 July 2008, Chicago, Illinois.

6. O. Kirmemis, A.Birturk, A User Controlled Content Based Movie Recommender with Explanation and Negative Feedback, In 4th International Confer-

ence on Web Information Systems and Technologies (WEBIST '08), 4-7 May 2008, Funchal, Portugal.

7. O. Kirmemis, A.Birturk, A Content Based Movie Recommendation System, in Expanding the Knowledge Economy: Issues, Applications, Case Studies, Paul Cunningham and Miriam Cunningham(Eds), IOS Press, 2007 Amsterdam, ISBN 978-1-58603-801-4; eChallenges 2007, 24-26 October 2007, The Hague, The Netherlands.

AWARD AND SCHOLARSHIP

- Ph.D. Fellowship by TUBITAK (2010-2015)