VILNIUS UNIVERSITY

JULIJA
PRAGARAUSKAITĖ

FREQUENT PATTERN ANALYSIS FOR DECISION MAKING IN BIG DATA

Doctoral Dissertation
Physical Sciences, Informatics (09 P)

Vilnius, 2013

Doctoral dissertation was prepared at Institute of Mathematics and Informatics of Vilnius University in 2008 – 2013.

Scientific Supervisor:

Prof. Dr. Habil. Gintautas Dzemyda (Vilnius University, Physical Sciences, Informatics – 09 P).

VILNIAUS UNIVERSITETAS

JULIJA
PRAGARAUSKAITĖ

DAŽNŲ SEKŲ ANALIZĖ SPRENDIMŲ PRIĖMIMUI LABAI DIDELĖSE DUOMENŲ BAZĖSE

Daktaro disertacija
Fiziniai mokslai, informatika (09 P)

Vilnius, 2013

Disertacija rengta 2008 – 2013 metais Vilniaus universiteto Matematikos ir informatikos institute.

Mokslinis vadovas:

prof. habil. dr. Gintautas Dzemyda (Vilniaus universitetas, fiziniai mokslai, informatika – 09 P).

Contents

Acknowledgments

First of all, I would like to thank my father Prof. Habil. Dr. Henrikas Pragarauskas who inspired me to start doctoral studies and supported me all the time by consulting, pushing forward my ideas and significantly increasing the quality of the first papers during my doctoral studies. It was with my deep sorrow to lose the closest family member in 2011 when he passed away suddenly from a heart attack. I am sure that he would be proud of me completing the work started even though he will never be able to give me his constructive comments regarding this thesis.

I want to express my big thanks to my scientific supervisor Prof. Dr. Gintautas Dzemyda for the patience, support and guidance of the research that finally evolved into this dissertation. He always spread the energy and always found time to meet me and have long discussions about new ideas, improvements and further guidance. I also want to express my special thanks to Dr. Juozas Gordevicius who provided a great support for me during this challenging period with providing a constructive feedback and asking critical questions that helped to significantly improve the quality of this thesis and my papers. I am also grateful for the constructive feedback and energy from other members of staff in System Analysis Department of Institute of Mathematics and Informatics.

I greatly appreciate the time and effort of the committee members and especially the reviewers Prof. Dr. Romas Baronas and Doc. Dr. Olga Kurasova who read and provided constructive feedback regarding this dissertation. Special thanks to Dr. Juozas Gordevicius, Prof. Dr. Remigijus Mikulevicius and Juan Madronero, who helped to proofread the thesis.

Finally, I wish to thank all my relatives and friends for all their support and patience during this challenging period of my life.

Abstract

Huge amounts of digital information are stored in the World today and the amount is increasing by quintillion bytes every day. Processing of this information using traditional techniques is becoming increasingly difficult. The challenges include capture, storage, search, transfer, analysis, and visualization of big datasets. 90% of the data stored in the World today have been created in the last two years alone and continue to grow at the same pace. However, these sheer amounts of data bring significant benefits and make it possible to do many things that could not be done previously: predict future behaviour of Internet users, diagnose diseases, identify crime suspects, forecast financial trends, etc.

Approximate (probabilistic) data mining algorithms are very important to efficiently deal with such amounts of information and to deliver the results in an acceptable timeframe required by various real-world applications. While approximate algorithms do not deliver precise results, the availability of theoretical estimations of their errors enables data analysts to decide if the precision offered by an approximate method satisfies the needs of the particular application. Exact data mining methods, on the contrary, tend to be slow and are best employed where the precise results are of the highest important.

This thesis focuses on several data mining tasks related to analysis of big data: frequent pattern mining and visual representation.

For mining frequent patterns in big data, three novel approximate methods are proposed and evaluated:

- Random Sampling Method (RSM) that creates a random sample from the original database, analyzes the sample and makes assumptions on the frequent and rare sequences in the original database based on the analysis results of the random sample. A significant benefit of this

method is that it is possible to provide a theoretical estimate of classification errors made by this method using standard statistical methods. Therefore, the proposed method can be employed without extensive empirical performance evaluations that are necessary for other state-of-the-art approximate methods.

- Multiple Re-sampling Method (MRM) is an improved version of RSM method with a re-sampling strategy. This method eliminates the class of intermediate sequences that has the highest probability to be incorrectly classified as frequent or rare sequences. This is done by performing the analysis of multiple random samples and making decision if the intermediate sequence is frequent or rare based on the results of the analysis of multiple samples.
- Markov Property Based Method (MPBM) that relies upon the Markov property. MPBM requires reading the original database at least several times (the number of times equals to the order of the Markov process) and then calculates the empirical frequencies of all higher level sequences using the Markov property. This method requires more computing time compared to RSM or MRM methods, as it reads the original database at least twice, whereas RSM and MRM methods do not read the entire original database at all, only the random sample.

The performance of proposed approximate methods in terms of the computing speed and precision was evaluated on real and artificial databases and compared with other existing state-of-the-art frequent pattern mining algorithms.

For visual representation of big data, the behaviour of Internet users was analyzed using geometric methods, multidimensional scaling and artificial neural networks. In visual representation of Internet users' behavioural data, SOM visualization showed the best results; however it requires some training and experience by a user, who is analyzing the data. Based on the

visualization, the decisions on the advertising strategy could be taken by a non-technical person who is managing the advertising campaigns.

List of Figures

List of Tables

Abbreviations and Acronyms

ApproxMAP	Approximate Multiple Alignment Pattern mining algorithm
DNA	Deoxyribonucleic acid (DNA) is a molecule that encodes the genetic instructions used in the development and functioning of all known living organisms and many viruses
Eclat	Equivalence class transformation algorithm
FP-growth	Frequent Pattern growth algorithm
FP-tree	Frequent Pattern tree used in Frequent Pattern growth algorithm
GSP	Generalized Sequence Patterns algorithm
MAFIA	Maximal frequent itemset algorithm for transactional databases
MaxMiner	Maximal frequent itemset mining algorithm
MDS	Multidimensional scaling
LAPIN	Last Position Induction algorithm
MPBM	Markov Property Based Method
MRM	Multiple Re-sampling Method
PRISM	Prime-Encoding Based Sequence Mining algorithm
ProMFS	Probabilistic algorithm for Mining Frequent Sequences
RSM	Random Sampling Method
sid	A sequence id in SPADE algorithm
eid	An event id in SPADE algorithm

SOM	Self Organizing Map
SPADE	Sequential Pattern Discovery using equivalence classes algorithm
SPAM	Sequential Pattern Mining algorithm

Introduction

Research Context and Motivation

The ever increasing amounts of digital information stored around the World today has the potential to bring significant benefits to people and make it possible to do many things that could not be done previously: forecast financial trends, diagnose and prevent diseases, identify crime suspects, etc. The amount of digital information increases enormously every year and according to the Moore's law, the processing power and storage capacity of computer chips double approximately every 18 months. The world's technological per-capita capacity to store information has roughly doubled every 40 months since the 1980s; as of 2012, every day 2.5 quintillion bytes of data were created (IBM review on what is big data, retrieved 2013). Keeping in mind the amount of raw data, today's algorithms and powerful computers can reveal new important insights that could previously remain hidden.

Frequent pattern mining is a very important task in data mining, especially in big data that consist of millions of records (Han, Kamber, 2006). The term "big data" is used very often recently in data mining and it defines a collection of data sets that are so large and complex that it becomes difficult to process them using on-hand database management tools or traditional data processing applications. The challenges include the capture of data, storage, search, sharing, transfer, analysis, and visualization. Data analysts regularly encounter limitations due to big data in many areas, including meteorology, genomics (Editorial review in Nature, 2008), complex physics simulations, biological data, environmental data, Internet search, finance and business informatics. Finding frequent patterns plays an essential role in mining associations, correlations, and many other interesting relationships among big data. Moreover, it helps in data classification, clustering, and other data mining tasks as well. Thus, frequent pattern mining has become an important data mining

task and attracted a number of different researches in frequent pattern mining. Exact frequent sequence mining methods deliver accurate results on frequent and rare sequences, however they typically require multiple passes over the original database and, if the database is large, it becomes a time consuming and expensive task. Approximate frequent sequence mining with an estimated probability of error is acceptable in many applications, e.g. marketing, internet user behaviour, stock market, biological databases, etc., because such algorithms are employed in scenarios where timely analysis result is more important than high precision. For example, if an investor can obtain all the approximate frequent sequences from her stock data quickly, then such approximate results might be sufficient to supplement the investor's own investment strategy and to allow the investor to take an optimal and profitable decision on a specific investment. On the other hand, if a stock investor attempts to extract all the accurate frequent sequences from the stock data, it would be very time-consuming and by the time a decision needs to be made, the best time for the investment might have passed. Combining different trading strategies with the knowledge of frequently occurring sequences might be a key to the trader's success; however this is typically a challenging task because frequent patterns have to be identified as quickly as possible. Firstly, the amount of data is typically so huge that it becomes difficult to both store and process it within an acceptable. Secondly, previously acquired knowledge may age as the time goes buy and lose its importance. Therefore, in financial markets the computing speed of finding frequent patterns in big data is very important as is using the approximate frequent sequence mining algorithms with an acceptable error threshold. This approach could help making quick decisions with higher profitability on the trading strategy. Frequent sequences have to be identified and presented to the trader as quickly as possible as every second is important to decide whether to open/exit the trading position and the delay could cause that the right moment of opening the trading position might have passed.

Approximate frequent sequence mining methods are much faster than exact methods because instead of doing multiple passes over the original database they analyze a much shorter sample of the original database or are using specific assumptions on the structure of the original database. In many cases finding an exact result in frequent sequence mining is not compatible with limited availability of resources and real time constraints, but an approximation of the exact result is enough for most purposes, e.g. biological data analysis, stock market analysis, behaviour analysis in the internet, etc. In the past decade there were a number of approximate frequent sequence mining algorithms proposed (ApproxMAP (Pei et al., 2003), ProMFS (Tumasonis, Dzemyda, 2004), approximate frequent sequence mining in data streams (Silvestri, Orlando, 2007)), that are much faster compared to exact frequent sequence mining algorithms.

All state-of-the-art approximate frequent sequence mining methods do not use a theoretical estimate of the probability of error made by the method when identifying frequent patterns in the original database. These methods provide only the empirical evidence based on extensive experiments and observations of algorithm results on different databases.

In this study we propose novel approximate frequent sequence mining methods with a theoretical estimate of the probability of error made by proposed methods when classifying sequences as frequent or rare. The performance of methods is compared to other approximate and exact frequent sequence mining methods and recommendations given on how to select parameters used in proposed methods to achieve more accurate results.

Problem Statement

Frequent sequence mining in big data is an important and yet a challenging data mining task. Frequent sequence mining has been analyzed for more than a

decade and has become an important component of many prediction or recommendation systems, e.g. predicting which web pages the online user is likely to visit or predicting which products the online user is likely to buy together.

All algorithms used for frequent sequence mining could be classified either as exact or approximate algorithms. Exact frequent sequence mining algorithms usually read the whole database many times, and if the database is very large, then frequent sequence mining is not compatible with limited availability of computer resources and real time constraints. Also in some cases exact methods cannot be used for continuous data streams, because the amount of data is typically so huge that it becomes difficult to store it and almost impossible to fully read it and process on time when the decision maker needs it. However in some cases where a precise result is required, then exact methods are irreplaceable by any approximate methods, because the precision is very important. For example, in biological databases an important task is similarity search and comparison between diseased and healthy tissues to identify critical differences between the two classes of genes. This can be done only using exact frequent pattern methods in order to retrieve the gene sequences from the two tissue classes and compare the frequently occurring patterns of each class. Usually sequences occurring more frequently in the diseased samples than in the healthy samples might indicate the genetic factors of the disease (Han, 2002).

Approximate methods are very popular in analyzing data where the computation speed of the algorithm is more important than the precision and some classification errors into frequent and rare sequences are acceptable. Trading in the Foreign Exchange is a good example in order to show the importance of quick approximate algorithms for frequent sequence mining. The knowledge of frequently occurring sequences is one of techniques being used in trading strategy and with the combination of other trading techniques the decisions in trading strategy could be taken when to open a trading

position. There is a huge amount of data captured every second in the financial markets, e.g. currency exchange data, stock market, etc. Traders often use technical analysis for forecasting the direction of prices through the study of past market data, primarily price and volume. There are many techniques in technical analysis, e.g. candlestick charting, Dow Theory, etc. and many traders combine elements from more than one technique. Some traders use a very subjective judgment to decide which pattern a price rate reflects at a given time and what the interpretation of that pattern should be. Others use a strictly mechanical approach to pattern identification and interpretation and search for archetypal price chart patterns, such as the well-known head and shoulders, double top/bottom reversal patterns or different market indicators, which are mathematical transformations of price. These patterns and indicators are used to help assessing whether an asset is trending, and if it is, then forecast of its direction. Immediate results from historical stock market data are needed for traders to estimate whether it is worth to enter or exit trading positions, therefore fast approximate algorithms are the key as none of exact algorithms can be used due to the continuous data stream. Any improvement in time taken for probabilistic algorithms is a big win, because every millisecond is very important to gain more profit in the trading strategy.

Various approximate sequence mining algorithms were proposed recently which are much faster than exact algorithms, however do not have the theoretical error estimation of algorithm results and this leads to a number of empirical tests being performed in order to get the empirical evidence of errors made identifying frequent sequences. Usually such algorithms are sensitive to the data that is used for finding frequent sequences and the experiment results might vary significantly depending on the data. The theoretical estimation of errors made identifying frequent sequences gives a huge advantage for data analysis because then it is known what precision might be used when finding frequent sequences in large databases.

Therefore, the goal of this study is to propose novel approximate frequent sequence mining algorithms with theoretically defined error probabilities and to compare their performance with other exact and approximate approaches.

In addition to frequent sequence mining, a visual representation technique for big data is proposed. It is very important for decision making when studying behaviour in the Internet users.

Tasks and Objectives of the Research

In this study the objectives are:

(1) create novel approximate frequent sequence mining algorithms for which theoretical estimation of induced errors can be made;

(2) propose an approach to visualise complex data, consisting of many records.

In pursuit of the above objectives, the specific tasks are:

- Study the existing frequent sequence mining algorithms (both exact and approximate);
- Propose novel approximate frequent sequence mining method with an estimate of the probability of error made by this method when classifying the sequences as frequent or rare;
- Construct of a sample of an original database for proposed approximate method;
- Propose random sampling based methods for mining frequent sequences with the estimated probabilities of errors made by these methods;
- Propose a method that relies upon the Markov property for approximately mining frequent sequences;

- Evaluate the performance of proposed approximate methods and compare the results with other exact and approximate frequent sequence mining approaches;
- Visually represent the analysis of behaviour of Internet user using various visualization methods.

Proposed Solutions and Contributions of Scientific Novelty

As a solution to the previously described problem the following three novel frequent sequence mining methods for dealing with big data and visualization of internet users' behaviour are proposed in this thesis:

- Random Sampling Method (RSM) with estimated classification errors using the central limit theorem;
- Multiple Re-sampling Method (MRM) which is an improved version of RSM method that includes re-sampling strategy;
- Markov Property Based Method (MPBM) for frequent sequence mining;
- Visual analysis of behaviour of Internet users.

All three proposed methods for frequent sequence mining are approximate and deliver approximate results in identifying frequent and rare sequences in the original database. All proposed methods are implemented and tested on real and artificial databases and their performance compared with other approximate and exact methods. Their theoretical error bounds are formulated and proved. The recommendation which parameters to choose by proposed methods is given to the reader in order to get more precise results using these methods.

For visual representation of big data, the behaviour of Internet users was analyzed using geometric methods, multidimensional scaling and artificial

neural networks. In visual representation of big data, SOM visualization showed the best results when visualizing the behaviour of Internet users; however it requires some training and experience by a user, who is analyzing the data. Based on the visualization, the decisions on the advertising strategy could be taken by a non-technical person who is managing the advertising campaigns.

In the following sections there is provided a detailed overview of the above tasks.

Defended Statements

1) Approximate (probabilistic) frequent sequence mining methods could deliver the acceptable results with right balance between the computing speed and precision when dealing with significantly growing amount of digital information in big data; whereas the exact frequent sequence mining methods work under the time constraints and cannot guarantee the required speed by various real-world applications.
2) Random Sampling Method (RSM) brings a significant benefit among existing approximate frequent sequence mining methods by providing a theoretical estimation of error probabilities. The error probabilities made by this method could be theoretically estimated using standard statistical methods. The theoretical estimation allows using the proposed method without any extensive empirical experiments used in other state-of-the-art methods to evaluate the performance of the method.
3) Random Sampling Method may be improved using a re-sampling strategy and eliminating the class of intermediate sequences as proposed in Multiple Re-sampling Method (MRM). The elimination of the class of the intermediate sequences delivers more precise results compared to Random

Sampling Method, however requires more computing time (as much as the number of re-samples analyzed).

4) Approximate frequent sequence mining methods that require reading the entire original database do not satisfy the time constraints required by many real-world applications. Therefore Markov Property Based Method (MPBM) proposed in this thesis, does not satisfy the time constraints as it requires reading the original database at least several times (the number of times equals to the order of the Markov process) and it delivers similar precision results as Multiple Re-sampling Method (MRM).
5) In the visual representation of Internet users' behavioural data, SOM visualization gives the best results; however it requires some training and experience by a user, who is analyzing the data.

Approbation

Published Papers

1. J. Pragarauskaitė, G. Dzemyda. Visual Decisions in the Analysis of Customer Online Shopping Behaviour. Nonlinear Analysis: Modelling and Control, Vol. 17, ISSN: 1392-5113, No. 3, p. 355–368, 2012.
2. J. Pragarauskaitė, G. Dzemyda. Markov Models in the analysis of frequent patterns in financial data. Informatica, Vol. 24, ISSN: 0868-4952, No. 1, p.87 – 102, 2013.
3. J. Pragarauskaitė, G. Dzemyda. Tikimybinis dažnų posekių paieškos algoritmas. Informacijos mokslai, Nr. 50, ISSN 1392-0561, p. 352-357, 2009.
4. J. Pragarauskaitė, G. Dzemyda. Probabilistic algorithm for mining frequent sequences. In Proceedings ASMDA 2011, Sapienza University of Rome, p. 1454-1460, Edizioni ETS, 2011.

5. J. Pragarauskaitė, G. Dzemyda. Tikimybinis dažnų posekių paieškos algoritmas. Lietuvos matematikos rinkinys, 51, p. 313–318, ISSN 0132-2818, 2010.

Conferences

International conferences

1. Probabilistic algorithm for mining frequent sequences. In the 10th International Conference on Probability and Mathematical Statistics, June 28 – July 2, 2010, Vilnius;
2. Markov models in the analysis of frequent pattern sequences. In the 16th International Conference on Mathematical Modelling and Analysis, May 25-28, 2011, Sigulda, Latvia;
3. Probabilistic algorithm with Markov for mining frequent sequences. In the 14th Conference of the ASMDA (Applied Stochastic Models and Data Analysis), June 7-10, 2011, Rome, Italy.

Regional conferences

1. Tikimybinis dažnų posekių paieškos algoritmas / Kompiuterininkų dienos - 2009,
   2009 09 25-26 d. Kaunas;
2. Tikimybinis dažnų posekių paieškos algoritmas / Lietuvos matematikų draugijos 51-oji konferencija, 2010 06 17-18 d., Šiauliai.

Outline of the Dissertation

In Chapter 1 an introduction to frequent sequence mining is provided, including the definition and related work in frequent sequence mining

algorithms. This chapter also shows the difference between frequent sequence and itemset mining.

In Chapter 2 detailed information about the most popular exact frequent sequence mining algorithms is presented, which also provides information about the most popular exact frequent sequence mining algorithms.

In Chapter 3 detailed information about the proposed novel approximate frequent sequence mining methods is presented.

Chapter 4 contains the performance studies of the proposed algorithms and the comparisons with other approximate and exact frequent sequence mining methods.

In Chapter 5 the methodology for analysis of the behaviour of Internet users is presented and examined.

Chapter 1: Frequent sequence mining: overview and problems

1.1. Introduction

Ever since the introduction of market basket analysis problem in 1993 (Agrawal et al., 1993) the issue of frequently appearing patterns has attracted numerous research efforts. Frequent sequence mining is a very important task in data mining and especially in big data, which consist of millions of records (Han, Kamber, 2006). All algorithms proposed for frequent sequence mining could be classified to exact or approximate (also known as probabilistic) algorithms:

- Exact algorithms for frequent sequence mining scan the entire original database and make multiple passes over it and if the database is large, then it is a time consuming and expensive task.
- Approximate algorithms for frequent sequence mining are faster than exact methods because instead of doing multiple passes over the original database they analyze a much shorter sample of the original database formed in a specific way and make assumptions on frequent and rare sequences in the original database based on the results of sample analysis.

Both exact and approximate algorithms are irreplaceable in various applications and the preferred algorithm is selected according to what is required by specific application: the computation speed or the accuracy of the results. Exact frequent sequence mining algorithms are very important in medicine, DNA, biology and other areas where the accurate answer is required; whereas approximate algorithms deliver quick approximate results in various applications where the estimated error probability is acceptable, e.g. internet data, marketing, finance, etc.

The approximate frequent sequence mining algorithms became even more relevant and popular for finding frequent sequences in big data, where the data is so large and complex that it becomes difficult to process it quickly using traditional data processing methods that read the entire database. Having in mind the challenges caused by increasing volume and variety of big data, it is obvious that in many areas, where the computation speed of the answer is critical, only approximate algorithms for frequent pattern mining could deliver desired quick results. It is also very important to have a theoretical estimation of the classification errors into frequent and rare sequences made by approximate frequent sequence mining algorithms and this way improve the veracity of decisions taken on the big data.

Another very important application of the approximate frequent sequence mining algorithms is continuous data streams, where new data arrives constantly and the volume of transactions can be infinite. Due to the fact that previously acquired knowledge may age as time goes by and lose its importance in data streams, approximate algorithms are needed for finding frequent sequences without storing the actual stream and ever consulting the old data. Having frequent sequences identified quickly in the continuous financial data stream, decisions with higher accuracy or profitability could be taken and are very important in different areas, e.g. financial markets. So frequent pattern mining algorithms in data streams can only guarantee an approximate result.

Frequent sequence mining studies can broadly be divided into two categories: frequent itemset mining and frequent sequence mining (Han et al., 2007). Mining of frequent itemsets focuses on finding the frequency of items being used together where the order of items is not important (Agrawal, Srikant, 1994; Brin et al., 1997; Han et al., 2007; Han et al., 2000, Park et al., 1995; Park et al., 1995; Sarawagi et al., 2000; Savasere et al., 1995; Zaki, 2000). The discovery of frequent itemsets is often called the discovery of association rules in the original database.

Frequent sequence mining, on the other hand, is concerned with the order in which the items arrive (Agrawal, Srikant, 1995; Ayres et al., 2002; Han et al., 2001; Srikant, Agrawal, 1996; Zaki, 2001) and the patterns can be itemsets, sequences, subtrees, subgraphs, etc. In the following chapters the definition and the difference between frequent sequence and itemset mining is provided as well as an overview of the existing algorithms for frequent sequence and itemset mining.

This thesis aims to propose approximate frequent sequence mining algorithms that have error probabilities defined by a theoretical estimate using standard statistical methods. To the best of our knowledge all previously proposed approximate algorithms (ApproxMAP, ProMFS) for frequent sequence mining provide only empirical evidence on their performance and reliability and they do not use a theoretical estimate of errors made when identifying frequent patterns in the original database. These methods provide only the empirical evidence based on extensive experiments and observations of algorithm results on different databases.

1.2. Definition of the frequent sequence and itemset

Frequent sequence mining focuses on finding frequent sub-sequences, where the order of sub-sequences is important. In order to find frequent sequences in the original database all the combinations of possible sub-sequences have to be generated and their frequencies in the original database identified.

Let the original database be a set $S = \{x_1, x_2, \dots, x_N\}$ whose elements x_i take M different values in a set $V = \{\alpha_1, \alpha_2, \dots, \alpha_M\}$. A sequence $(\alpha_{i_1}, \dots, \alpha_{i_m})$ is called frequent if (# is a number):

$$p(\alpha_{i_1}, \dots, \alpha_{i_k}) = \frac{1}{N-m+1} \#\{j \in \{1, \dots, N-m+1\}: x_j = \alpha_{i_1}, x_{j+1} = \alpha_{i_2}, \dots, x_{j+m-1} = \alpha_{i_m}\} \geq \varepsilon,$$

where $\varepsilon\, \epsilon\, (0,1)$ is a user-specified threshold, called a minimum support threshold.

In contrast to the frequent sequence mining, where the order of sub-sequences is important, frequent itemset mining focuses on finding itemsets that are frequent in many transactions and the order of items appearing in the itemset is not important. Intuitively in the frequent itemset mining a set of items that appears in many baskets is said to be frequent.

Let $I = \{i_1, i_2, \ldots, i_n\}$ be a set of all items in specific transaction and T is a database that contains all shopping baskets (transactions). A k-itemset, which consists of k items from I, is frequent if it occurs in a transaction database T no lower than ε times, where ε is a user-specified minimum support threshold.

The following chapters contain the overview about the existing algorithms for frequent sequence and itemset mining.

1.3. Frequent itemset mining

The discovery of frequent itemsets in large database is often viewed as the discovery of association rules and aims to find itemsets that are appearing in the original database not less than a user-specified minimum support threshold ε. The first and the most famous application of frequent itemset mining is market-basket exercise that was introduced in 1994, where it is important to find the sets of items that appear in many various baskets or so called transactions. Each basket consists of a set of items, where usually the number of items in a basket is relatively small – much smaller than the total number of items, whereas the number of baskets is usually assumed to be very large, much bigger than what can fit in the main memory.

There are 3 main basic methodologies used in frequent itemset mining algorithms (Han et al., 2007):

1. Apriori;
2. FP-growth;
3. Eclat.

Apriori method was proposed in 1994 by R. Agrawal et al. and it relies upon a downward closure property, called Apriori, which states that an itemset is frequent only if all of its sub-items are frequent too. This is the essence of the Apriori algorithm (Agrawal, Srikant, 1994). The Apriori algorithm scans the entire database multiple times and generates candidate itemsets of length k from itemsets of length $k-1$. Then it prunes the candidates that have infrequent sub-items. After the Apriori algorithm was introduced, there were a number of improvements and extensions proposed, e.g. hashing technique (Park et al., 1995), partitioning technique (Park et al., 1995), dynamic itemset counting (Brin et al., 1997), and other (Sarawagi et al., 2000; Savasere et al., 1995). Even though in many cases the Apriori algorithm significantly reduces the generation of the candidates, it can suffer from 2 nontrivial costs: (1) generating a huge number of candidate sets, and (2) repeatedly scanning the entire database as many times as the longest frequent itemset and pruning the list of frequent itemsets from the list of generated candidates. For these two reasons the Apriori algorithm could not be used for mining frequent sequences in data streams and cannot generally be efficiently used in big data.

Han et al. in 2000 proposed FP-growth (Frequent Pattern growth) algorithm (Han et al., 2000), which mines the complete set of frequent itemsets without candidate generation as in the Apriori algorithm. The FP-growth algorithm uses frequent pattern tree (FP-tree) structure, which is an extended prefix-tree structure for storing compressed, crucial quantitative information about frequent patterns. Only frequent items which consist of 1 element have nodes in the tree, and the tree nodes are arranged in such a way that more frequently occurring nodes will have better chances of sharing nodes than less frequently

occurring ones. Further candidate generation is achieved via concatenation of the suffix pattern with the new ones generated from an FP-tree. The search technique employed in mining is a partitioning-based, divide-and-conquer method rather than Apriori-like bottom-up generation of frequent itemsets combinations. Moreover, it transforms the problem of finding long frequent patterns to looking for shorter ones and then concatenating the suffix. There are many alternatives and extensions to the FP-growth approach, including depth-first generation of frequent itemsets (Agarwal et al., 2001); H-Mine (Pei et al., 2001) which explores a hyper-structure mining of frequent patterns; building alternative trees; exploring top-down and bottom-up traversal of such trees in pattern-growth mining by Liu et al. (Liu et al., 2002; Liu et al., 2003); and an array-based implementation of prefix-tree-structure for efficient pattern growth mining (Grahne, Zhu, 2003). The performance study showed that the FP-growth algorithm is efficient and scalable for mining both long and short frequent itemsets, and is about an order of magnitude faster than the Apriori algorithm. However extensive experiments with large databases showed that FP-growth algorithm is costly in terms of memory use and this encouraged further research and algorithms proposed for frequent itemset mining.

Eclat (Equivalence CLASS Transformation) algorithm was proposed by M. J. Zaki in 2000 (Zaki, 2000) and it uses a vertical data format, whereas both Apriori and FP-growth algorithms use a horizontal data format. The horizontal data format consists of a list of transactions, where each transaction has an identifier followed by a list of items in that transaction, whereas the vertical data format associates each itemset with a list of transactions in which it occurs. First of all, the algorithm involves the computation of the frequent itemsets of size one and two. Further frequent itemsets are generated by intersecting the tid-lists (transaction-id-lists) of all distinct pairs of atoms and checking the cardinality of the resulting tid-list. A recursive procedure call is made with those itemsets found to be frequent at the current level. This process is repeated until all frequent itemsets have been enumerated. Eclat algorithm

scans the database only once to discover frequent itemsets, therefore it demonstrates much better results than the Apriori or FP-tree algorithms.

This thesis focuses on the frequent sequence mining in large databases, therefore the following chapters in the thesis detail existing and proposed algorithms for frequent sequence mining rather than frequent itemset mining.

1.4. Frequent sequence mining algorithms

1.4.1. Exact algorithms for frequent sequence mining

Frequent sequence mining (also known as sequential pattern mining) focuses on finding frequent sub-sequences, where the order of sub-sequences is important. While in frequent itemset mining only the combinations of the items has to be generated, in case of frequent sequence mining the variations of the combinations of the items has to be generated. Frequent pattern mining was first introduced by Agrawal et al. in 1995 (Agrawal, Srikant, 1995). In 1996 the same authors proposed GSP (Generalized Sequence Patterns) algorithm (Srikant, Agrawal, 1996), which was a representative of Apriori-based algorithms and implemented a candidate generation, testing and pruning approach.

In 2001 M. J. Zaki proposed a SPADE algorithm (Zaki, 2001), which uses a vertical format to store the sequence patterns, which reduced the number of database scans. However both algorithms GSP and SPADE use the Apriori pruning principle and both generate a large set of candidates for longer sequences; therefore due to multiple passes over the database, these algorithms could hardly be used in big data or in continuous data streams.

A different approach - PrefixSpan algorithm was proposed by J. Han et al. (Han et al., 2001) which works in a divide-conquer way and each sequential

pattern is treated as a prefix and the complete set of sequential patterns can be partitioned into different subsets according to the different prefix and the mining of sequential patterns is done recursively. According to the algorithms performance comparison, PrefixSpan shows overall best performance, then SPADE outperforms GSP, which is in the last place.

SPAM algorithm (Sequential PAttern Mining) (Ayres et al., 2002) for mining sequential patterns was proposed by J. Ayres in 2002. The algorithm uses the first depth-first search strategy for mining sequential patterns and a vertical bitmap representation of the database allowing for efficient support counting. PrefixSpan outperforms SPAM slightly on very small datasets, but on large datasets SPAM outperforms PrefixSpan and SPADE by over an order of magnitude. However SPAM assumes that the entire database (and all data structures used for the algorithm) completely fit into main memory and therefore the algorithm could not be applied to frequent sequence mining in data streams.

LAPIN algorithm (LAst Position INduction algorithm) (Yang et al., 2006) was proposed in 2006 by Z. Yang et al. has outperformed all previous existing algorithms, especially on long, dense datasets. The algorithm is based on the simple idea that the last position of an item, α, is the key to judging whether or not a frequent k-length sequential pattern can be extended to be a frequent (k+1)-length pattern by appending the item α to it.

In 2007 Gouda et al. proposed PRISM algorithm (PRIme-Encoding Based Sequence Mining) (Guoda et al., 2007; Guoda et al., 2010), which utilizes a vertical approach for enumeration and support counting, based on the prime block encoding, which in turn is based on prime factorization theory. According to the performance study on both synthetic and real datasets, PRISM algorithm outperforms SPADE, PrefixSpan and SPAM algorithms at least by an order of magnitude.

All mentioned algorithms for frequent sequence mining are exact algorithms and require reading the entire original database multiple times. If the database is huge, these algorithms are very time-consuming and might require more time to find frequent sequences than it is acceptable in the specific real-world application where the algorithm is being used. For example in the financial markets by the time a decision could be made by the trader, the best time for investment might have passed if the algorithm takes longer than acceptable time threshold set by the trader. In order to satisfy such time constraints, the efficient approximate frequent sequence mining methods are being developed.

The detailed description of GSP, SPADE, PrefixSpan, SPAM, LAPIN and PRISM algorithms is provided in the Chapter 2.

1.4.2. Approximate algorithms for frequent sequence mining

Approximate methods are faster than exact methods because instead of doing multiple passes over the original database they analyze a much shorter sample of the original database and make specific assumptions on the frequent and rare sequences in the original database.

Recently, some methods for mining approximate frequent sequences were proposed: ApproxMAP algorithm (Pei et al., 2003) was proposed by J. Pei et al. in 2003. The idea of ApproxMAP algorithm is that, instead of finding exact patterns, the algorithm identifies patterns approximately shared by many sequences. ApproxMAP algorithm uses a cluster and multiple alignment based approach, which works in two steps: (1) the sequences in the original database are clustered based on similarity. Sequences in the same cluster may approximately follow some similar patterns. (2) the longest approximate sequential pattern for each cluster is generated (so called the consensus pattern) and in order to extract consensus patterns, a weighted sequence is derived for

each cluster using multiple alignments to compress the sequential pattern information in the cluster. And then the longest consensus pattern best representing the cluster is generated from the weighted sequence.

Another approach is being proposed in approximate ProMFS algorithm by R. Tumasonis et al. in 2004 (Tumasonis, Dzemyda, 2004), which is based on the estimated probabilistic-statistical characteristics of the elements appearance in the sequence and their order. The algorithm builds a new much shorter sequence, analyses it using the exact GSP algorithm (Srikant, Agrawal, 1996) and makes assumptions on the frequent and rare sequences in the original database based on the results of analysis of the shorter original database sample. The algorithm is purely based on the empirical analysis and is not based on the theoretical proof using the statistical methods.

Approximate Random Sampling Method (RSM) proposed in 2011 by the author of this thesis that analyzes a random sample of the original database and makes assumptions on the frequent and rare sequences in the original database according to the random sample analysis results. The method uses the exact GSP algorithm for analysing a random sample of the original database. The probability of errors made by the approximate RSM method is estimated using the statistical methods.

An improvement to Random Sampling Method (RSM) was proposed by the author of the thesis that is called Multiple Re-sampling Method (MRM). MRM method includes re-sampling strategy and this way decreases the number of sequences that might be incorrectly classified when determining if the sequence is frequent or rare.

Another novel approximate Markov Property Based Method (MPBM) (Pragarauskaitė, Dzemyda, 2013) for mining frequent sequences is proposed in this thesis. MPBM method makes only several passes over the original database and the frequency of higher level sequences in the original database is

calculated using the Markov property without reading the entire original database again.

The Chapter 3 contains a detailed description of the following approximate frequent sequence mining algorithms: ApproxMAP, ProMFS, RSM, MRM and MPBM.

1.5. Frequent sequence mining in data streams

The approximate frequent sequence mining algorithms became even more relevant and popular for finding frequent sequences in continuous data streams, where new data arrives constantly. Due to the fact that previously acquired knowledge may age as time goes by and lose its importance in data streams, the approximate algorithms are needed for finding frequent sequences without storing the actual stream and ever consulting the old data. Having frequent sequences identified quickly in the continuous financial data stream, decisions with higher accuracy or profitability could be taken and are very important in financial markets.

The existing frequent sequence mining algorithms essentially differ in the data structures used to index the database to facilitate the fast enumeration. The existing algorithms utilize three main approaches to sequence mining: horizontal (Agrawal, Srikant, 1995; Srikant, Agrawal, 1996), vertical (Ayres et al., 2002; Zaki, 2001) and projection based (Han et al., 2001). All these frequent sequence mining algorithms typically require multiple passes over the dataset. However in a streaming environment, a mining algorithm must take only a single pass over the data (Babcock et al., 2002).

In a data stream, transactions arrive continuously and the volume of transactions can be infinite. So frequent pattern mining algorithms in data streams can only guarantee an approximate result. The challenges of frequent pattern mining in data streams are that it costs much more space to generate an

approximate answer set in data streams; also the pruning process and the generation of new frequent patterns is a very compute-intensive task (Babnik et al., 2007). Consequently, keeping up pace with the high speed data streams can be a very difficult frequent pattern mining task.

Reading the entire data stream for mining frequent patterns is a challenging and expensive task (Giannella et al., 2003). As it is too expensive to read the entire data stream each time, the popular approach for mining frequent sequences in data streams is mining over the sliding window. Formally a data stream D can be defined as a sequence of transactions,

$$D = (t_1, t_2, \dots, t_i, \dots)$$

where t_i is the i-th arrived transaction. To process and mine data streams, different window models are often used. A window is a subsequence between i-th and j-th arrived transactions, denoted as

$$W[i, j] = (t_i, t_{i+1}, \dots, t_j), i \leq j.$$

The sliding window is a window that slides together with the time. Given the length of the sliding window s and the current time t, the mining of frequent sequences would be performed in the window $W[t - w + 1, t]$.

The damped window model also assigns weights to the recently arrived transactions, meaning that recent transactions are more important than previous ones.

Another approach is sampling, which refers to the process of probabilistic choice of a data item to be processed or not. Boundaries of the error rate of the computation could be estimated using statistical methods. So the probabilistic frequent sequence mining algorithms can be performed on the data streams by sampling the data within the sliding window of the data stream.

Random Sampling Method (RSM) proposed by the author of the thesis could also be used on the data stream using a large sliding window and the re-

sampling technique would give even more precise results on identifying frequent and rare sequences. Such approach is robust to the size of the input and thus allows very large amounts of data to be processed.

1.6. Conclusions

There are various exact and approximate algorithms used for frequent sequence and itemset mining, however in order to properly select the algorithm that is best suited for the particular real-world application, it is very important to know what is more important: the precision of results or the computation speed of delivering results. Exact algorithms for mining frequent sequences are irreplaceable in some applications, e.g. genetics, biology, medicine, etc., that require a precise answer on frequent and rare sequences. The exact GSP algorithm that works according to the Apriori principle was widely used in many different studies and algorithms since 1994 and various improvements were proposed to achieve better performance of the Apriori method to deliver quicker results. Exact PRISM and LAPIN algorithms for frequent sequence mining outperform all the other exact algorithms proposed in the past decade: GSP, PrefixSpan, SPADE and SPAM algorithms and could be used in a combination with the approximate methods proposed in this thesis.

There are many areas where the computing speed of the algorithm is critical compared to the precision, e.g. finance, Internet users' behaviour, marketing, etc., therefore the approximate methods for frequent sequence mining are irreplaceable. For data streams the exact algorithms can hardly be applied at all due to the fact that the data is arriving continuously into the data stream and storing the actual stream or ever consulting the old data does not make sense.

The importance of approximate algorithms is obvious in many real-world applications that require an immediate answer on the frequent and rare sequences. Approximate methods proposed in this study not only have good

performance results in terms of the computation speed of the method, but also have statistical estimation of errors made by the proposed approximate methods. This gives a significant benefit as the estimation of algorithm performance in terms of precision does not require extensive empirical tests and are not robust to the data analyzed using these algorithms.

Chapter 2: Exact algorithms for mining frequent sequences

2.1. Introduction and overview of algorithms

Exact frequent sequence mining algorithms deliver precise results on the classification of frequent and rare sequences and their true frequencies in the original database; however they require reading the entire original database at least several times. In big data or very large databases the exact frequent sequence mining algorithms could be very time-consuming and hardly could be used in the areas, where the computation speed of the answer is more important than the precision, e.g. marketing, behaviour of Internet users, stock market, biological databases, etc. In stock market any delays when identifying frequent and rare sequences used by the trading strategy might end up that by the time a decision could be made by the trader, the best time for the investment might have passed.

This chapter provides an overview of the exact frequent sequence mining algorithms and describes the state-of-the-art exact frequent sequence mining algorithms in details. Some approximate frequent sequence mining algorithms use exact algorithms to analyse a much shorter sample of the original database (ProMFS (Tumasonis, Dzemyda, 2004), RSM (Pragarauskaitė, Dzemyda, 2011), MRM) and make the decisions on the frequent and rare sequences in the original database based on the results of analysis of the shorter original database sample.

GSP (Generalized Sequence Patterns) algorithm (Srikant, Agrawal, 1996) was proposed by R. Agrawal and R. Srikant in 1996, which is a representative of Apriori-based algorithms and implements a candidate generation, testing and pruning approach. GSP became a well known and widely used algorithm in the frequent sequential pattern mining and widely reused in many other algorithms for mining frequent patterns.

SPADE algorithm (Zaki, 2001) was proposed by M. J. Zaki in 2001 which uses a vertical format to store the sequence patterns and reduces the number of database scans. Both GSP and SPADE algorithms use Apriori pruning and both generate a large set of candidates for longer sequences and make multiple passes over the original database.

A different approach - PrefixSpan algorithm (Han et al., 2001) was proposed by J. Pei et al. in 2001 which works in a divide-conquer way and each sequential pattern is treated as a prefix and the complete set of sequential patterns can be partitioned into different subsets according to the different prefix and the mining of sequential patterns is done recursively. According to the algorithms performance comparison, PrefixSpan shows overall best performance, then SPADE outperforms GSP, which is in the last place.

SPAM algorithm (Sequential Pattern Mining) (Ayres et al., 2002) was proposed by J. Ayres et al. in 2002 for mining sequential patterns that uses the first depth-first search strategy for mining sequential patterns and a vertical bitmap representation of the database allowing for efficient support counting. PrefixSpan outperforms SPAM slightly on very small datasets, but on large datasets SPAM outperforms PrefixSpan and SPADE by over an order of magnitude.

LAPIN (Last Position Induction algorithm) (Yang et al., 2006) was proposed by Z. Yang et al. in 2006 for mining sequential patterns that offers a solution for 2 main challenges that all previous exact algorithms were suffering: large search spaces and the ineffectiveness in handling dense datasets. LAPIN algorithm largely reduces the search space during the mining process, and is very effective in mining dense datasets.

PRISM algorithm (Prime-Encoding Based Sequence Mining) (Guoda et al., 2007; Gouda et al., 2010) was proposed by K. Gouda et al. in 2007 which utilizes a vertical approach for enumeration and support counting, based on the prime block encoding, which in turn is based on prime factorization theory.

According to the performance study on both synthetic and real datasets, Gouda et al. shows that PRISM outperforms SPADE, PrefixSpan and SPAM algorithms at least by an order of magnitude.

In the following chapters the most popular exact frequent sequence mining algorithms are described: GSP, SPADE, PrefixSpan, SPAM, LAPIN and PRISM algorithms.

2.2. Generalized Sequential Pattern (GSP) algorithm

GSP (Generalized Sequential Pattern) algorithm (Srikant, Agrawal, 1996) is an exact algorithm to determine frequent sequences in the original database according to the user-specified minimum support threshold $\varepsilon \in (0,1)$. GSP algorithm works according to the Apriori downward closure principle, where the candidate sequences of length k are generated from the shorter length $k-1$ sub-sequences and then the candidate sequences that have infrequent sub-sequences are pruned. The detailed description of GSP algorithm is given below.

Let the original database be a set $S = \{x_1, x_2, \dots, x_N\}$ whose elements x_i take M different values in a set $V = \{\alpha_1, \alpha_2, \dots, \alpha_M\}$. A sequence $(\alpha_{i_1}, \dots, \alpha_{i_m})$ is called frequent if (# is a number):

$$p(\alpha_{i_1}, \dots, \alpha_{i_k}) = \frac{1}{N-m+1} \#\{j \in \{1, \dots, N-m+1\}: x_j = \alpha_{i_1}, x_{j+1} = \alpha_{i_2}, \dots, x_{j+m-1} = \alpha_{i_m}\} \geq \varepsilon,$$

where $\varepsilon \, \epsilon \, (0,1)$ is a user-specified threshold, called a minimum support threshold.

Otherwise if $p(a_{i_1}, \dots, a_{i_k}) < \varepsilon$, then the sequence $a_{i_1}, \dots, a_{i_k}$ is considered to be rare. GSP algorithm makes multiple passes over the original database and determines which sequences are rare and then eliminates them from the

candidates list. During the first pass over the original database GSP algorithm reads all first level (one element) sequences $a_1, a_2, \dots, a_m$ and determines which sequences are frequent according to the minimum support threshold ε. Then GSP algorithm generates the second level (two elements) sequences $a_1a_1, a_1a_2, \dots, a_1a_n, a_2a_1, \dots, a_2a_n, \dots, a_na_1, \dots, a_na_n$ from already identified first level frequent sequences excluding rare subsequences from the first level. It is obvious that if the sequence is rare, then all higher sequences that consist of this sequence are rare too, e.g. if the subsequence a_1a_2 is rare, then sequences $a_1a_2a_1, a_2a_1a_2$, etc. are rare too. Further passes over the original database are done in the similar way until there are no frequent sequences to be used for generating candidates in the higher level.

After GSP algorithm was proposed in 1996, there were a number of improvements and extensions proposed that improve the performance of GSP algorithm, e.g. hashing technique (Park et al., 1995), partitioning technique (Park et al., 1995), dynamic itemset counting (Brin et al., 1997), and other (Sarawagi et al., 2000; Savasere et al., 1995). Even though many of proposed improvements significantly reduce the size of candidates generated using the Apriori principle, however all these improvements for GSP algorithm still suffer from 2 nontrivial costs:

1. Generating a huge number of candidate sequences using Apriori principle;
2. Repeatedly scanning the original database and checking the candidates by pattern matching and then pruning the candidates. This requires multiple database scans, as many as the length of the longest frequent sequence.

Due to above mentioned drawbacks GSP algorithm and its modifications are costly and very time-consuming compared to other exact frequent sequence mining algorithms SPADE, SPAM, PrefixSpan, LAPIN or PRISM. This

results that GSP algorithm could hardly be used for mining frequent sequences in big data or data streams.

2.3. Sequential Pattern Discovery using Equivalence classes (SPADE) algorithm

SPADE algorithm (Zaki, 2001) was proposed by M. J. Zaki in 2001, which uses a vertical format to store the sequence patterns and reduces the number of the original database scans compared to GSP algorithm. SPADE algorithm utilizes combinatorial properties to decompose the original problem of finding frequent patterns into smaller sub-problems that can be independently solved in the main-memory using efficient lattice search techniques and simple join operations. Using SPADE algorithm all frequent sequences are discovered in three database scans, which is a significant improvement compared to GSP algorithm. The key features of SPADE algorithm are as follows:

- Using a vertical *id*-list database format, where SPADE algorithm associates with each sequence a list of objects in which it occurs, along with the time-stamps. This way all frequent sequences are enumerated via simple id-list intersections.
- Using a lattice-theoretic approach to decompose the original search space (lattice) into smaller pieces (sub-lattices) which can be processed independently in the main-memory. Such approach usually requires three database scans.
- Using two different search strategies for enumerating the frequent sequences within each sub-lattice: breadth-first and depth-first search approach.

M. J. Zaki defines the sequence α as an ordered list of events denoted as $\alpha_1 \rightarrow \alpha_2 \rightarrow \cdots \rightarrow \alpha_q$, where α_i is an event. Consider the original database D consists of a collection of input-sequences where each of them has a unique

identifier *sid* (sequence-id) and each event in a given input-sequence also has a unique identifier called *eid* (event-id). Given a user-specified minimum support threshold ε, it is said that a sequence is frequent if it occurs more than minimum support times. Assume that the original database D consists of the following input-sequences as displayed in Figure 1.

DATABASE

SID	Time (EID)	Items
1	10	C D
1	15	A B C
1	20	A B F
1	25	A C D F
2	15	A B F
2	20	E
3	10	A B F
4	10	D G H
4	20	B F
4	25	A G H

FREQUENT SEQUENCES

Frequent 1-Sequences

A	4
B	4
D	2
F	4

Frequent 2-Sequences

AB	3
AF	3
B->A	2
BF	4
D->A	2
D->B	2
D->F	2
F->A	2

Frequent 3-Sequences

ABF	3
BF->A	2
D->BF	2
D->B->A	2
D->F->A	2

Frequent 4-Sequences

D->BF->A	2

Figure 1. Original input-sequence database for SPADE algorithm

As an input SPADE algorithm takes the user-specified minimum support threshold ε and the original database D; then does the following steps:

- Computing $F_1 = \{$frequent 1-element sequences$\}$. Due to the vertical *id*-list database, all frequent 1-element sequences can be computed in a single database scan. For each database item, the algorithm reads item's *id*-list from the disk into the memory and scans the *id*-list, incrementing the support for each new *sid* encountered.
- Computing $F_2 = \{$frequent 2-elements sequences$\}$. In this step the transformation from the horizontal format to the vertical format is being

performed. For each item i, the algorithm reads it's *id*-list into the memory. For each sequence and event pair (sid, eid), SPADE algorithm inserts (i, e) in the list for input-sequence *s*. For example, consider the *id*-list for item A shown in Figure 2 below. SPADE algorithm scans the first pair (1,15) and then insert (A,15) in the list for input-sequence *sid* 1. Figure 3 shows the complete horizontal database recovered from the vertical item *id*-lists in Figure 2.

A	
SID	EID
1	15
1	20
1	25
2	15
3	10
4	25

B	
SID	EID
1	15
1	20
2	15
3	10
4	20

D	
SID	EID
1	10
1	25
4	10

F	
SID	EID
1	20
1	25
2	15
3	10
4	20

Figure 2. *id*-lists in SPADE algorithm

sid	$(item, eid)$ pairs
1	(A 15) (A 20) (A 25) (B 15) (B 20) (C 10) (C 15) (C 25) (D 10) (D 25) (F 20) (F 25)
2	(A 15) (B 15) (E 20) (F 15)
3	(A 10) (B 10) (F 10)
4	(A 25) (B 20) (D 10) (F 20) (G 10) (G 25) (H 10) (H 25)

Figure 3. Vertical to horizontal database recovery in SPADE algorithm

The computation from the recovered horizontal database is straight-forward: a list of all 2-elements sequences in the list for each *sid* is formed and updated counts in a 2-dimensional array indexed by the frequent items.

- Enumerating frequent sequences of a class. This procedure takes a set of atoms of sub-lattice *S* along with their *id*-lists as an input. Frequent sequences are generated by joining the *id*-lists of all pairs of atoms and checking the cardinality of the resulting *id*-list against

the minimum support threshold ε. The sequences found to be frequent at the current level form the atoms of classes for the next level. This recursive process is repeated until all frequent sequences have been enumerated.

SPADE algorithm minimizes the computation time by reducing database scans compared to GSP algorithm. The algorithm performance study shows that SPADE algorithm outperforms previous approaches of GSP algorithm and its modifications by a factor of two. Experiments were performed on a 100MHz MIPS processor with 256MB main memory running IRIX 6.2, with non-local 2GB disk. The synthetic datasets were used from the publicly available dataset generation code from the IBM Quest data mining project (Quest project, 2013). These datasets mimic real-world transactions where people buy a sequence of sets of items. The number of data-sequences was set to 200000. Figure 4 shows the computing time comparison between GSP and SPADE algorithms.

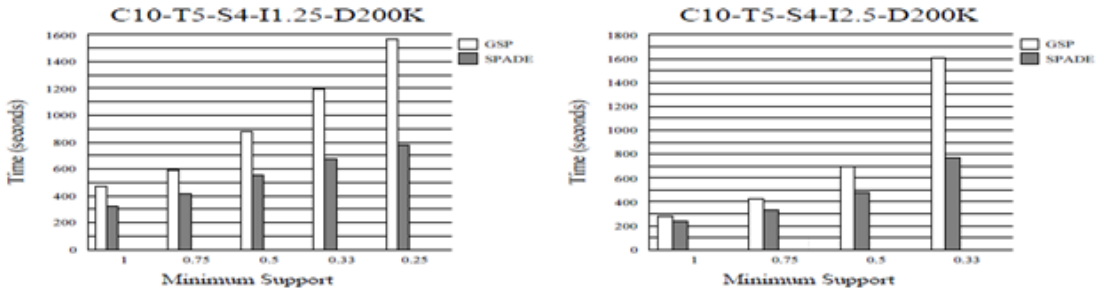


Figure 4. Performance Comparison of GSP and SPADE algorithms on Synthetic Datasets.

Both GSP and SPADE algorithms use Apriori principle including the pruning procedure, so they both generate a large set of higher level candidate sequences. Due to this and also the multiple passes over the original database, SPADE and GSP algorithms have serious limitations for using them in big data or data streams where the data arrives continuously and cannot fit into the main memory.

2.4. Prefix-projected Sequential pattern mining (PrefixSpan) algorithm

PrefixSpan (Prefix-projected Sequential pattern mining) algorithm (Han et al., 2001) was proposed by J. Han, J. Pei et al. in 2001. The algorithm which mines the complete set of patterns but greatly reduces the efforts of candidate generation. The key of PrefixSpan algorithm is to examine only the prefix subsequences and project only their corresponding postfix subsequences into projected databases. In each projected database sequential patterns are grown by exploring only local frequent patterns. To further improve the efficiency of PrefixSpan algorithm, two kinds of database projections are explored: level-by-level projection and bi-level projection. The performance study shows that bi-level projection has better performance when the database is large, and level-by-level-projection speeds up the processing substantially when the projected databases can fit in the memory. PrefixSpan algorithm has the following definitions.

Definition 1 (Prefix). Suppose that all items in an element are listed alphabetically. Given a sequence $\alpha = \langle e_1 e_2 \dots e_n \rangle$: a sequence $\beta = \langle e'_1 e'_2 \dots e'_m \rangle (m \leq n)$ is called a prefix of α if (1) $e'_i = e_i$ for $(i \leq m-1)$; (2) $e'_m \subseteq e_m$; and (3) all the items in $(e_m - e'_m)$ are alphabetically after those in e'_m.

Definition 2 (Projection). Given sequences α and β such that β is a subsequence of α. A subsequence α' of sequence α is called a projection of α with regard to prefix β if and only if (1) α' has prefix β and (2) there exists no proper super-sequence α'' of α' such that α'' is a subsequence of α and also has prefix β.

Definition 3 (Postfix). Let $\alpha' = \langle e_1 e_2 \dots e_n \rangle$ be the projection of α with regard to prefix $\beta = \langle e_1 e_2 \dots e_{m-1} e'_m \rangle (m \leq n)$. Sequence $\gamma = \langle e''_m e_{m+1} \dots e_n \rangle$ is called the postfix of α with regard to prefix β, denoted as $\gamma = \alpha / \beta$, where

$e''_m = (e_m - e'_m)^2$. We also denote $\alpha = \beta \cdot \gamma$. If β is not a subsequence of α with regard to β are empty.

Table 1. A sequence database in PrefixSpan algorithm

Sequence_id	Sequence
10	$\langle a(abc)(ac)d(cf)\rangle$
20	$\langle (ad)c(bc)(ae)\rangle$
30	$\langle (ef)(ab)(df)cb\rangle$
40	$\langle eg(af)cbc\rangle$

Let the running database be a sequence database S given in the Table 1 and the user-specificied min_support $= 2$. The set of items in the database S is $\{a, b, c, d, e, f, g\}$. The sequential patterns in S can be mined by a prefix-projection method in the following steps.

- **Step 1:** Find 1-element sequential patterns. Scan the sequence database S once in order to find all frequent items in the sequences. Each of these frequent items has a length of 1-element sequential pattern. The frequent items in the sequences are as follows: $\langle a\rangle: 4, \langle b\rangle: 4, \langle c\rangle: 4, \langle d\rangle: 3, \langle e\rangle: 3, \langle f\rangle: 3,$ where $\langle pattern\rangle: count$ represents the pattern and its associated support count.
- **Step 2:** Divide the search space. The complete set of sequential patterns can be partitioned into the following six subsets according to the six prefixes: (1) the ones having prefix $\langle a\rangle$; ...; and (6) the ones having prefix $\langle f\rangle$.
- **Step 3:** Find subsets of sequential patterns. The subsets of sequential patterns can be mined by constructing corresponding projected databases and mine each recursively. The projected databases as well as sequential patterns found in S are listed in Table 2, while the mining process is explained as follows:

Table 2. Projected databases and sequential patterns in PrefixSpan algorithm

Prefix	Projected (postfix) database	Sequential patterns
$\langle a \rangle$	$\langle (\_bc)d(cf) \rangle$, $\langle (\_d)c(bc)(ae) \rangle$, $\langle (\_b)(df)cb \rangle$, $\langle (\_f)cbc \rangle$	$\langle a \rangle$, $\langle aa \rangle$, $\langle ab \rangle$, $\langle a(bc) \rangle$, $\langle a(bc)a \rangle$, $\langle aba \rangle$, $\langle abc \rangle$, $\langle (ab) \rangle$, $\langle (ab)c \rangle$, $\langle (ab)d \rangle$, $\langle (ab)f \rangle$, $\langle (ab)dc \rangle$, $\langle ac \rangle$, $\langle aca \rangle$, $\langle acb \rangle$, $\langle acc \rangle$, $\langle ad \rangle$, $\langle adc \rangle$, $\langle af \rangle$
$\langle b \rangle$	$\langle (\_c)(ac)d(cf) \rangle$, $\langle (\_c)(ae) \rangle$, $\langle (df)cb \rangle$, $\langle c \rangle$	$\langle b \rangle$, $\langle ba \rangle$, $\langle bc \rangle$, $\langle (bc) \rangle$, $\langle (bc)a \rangle$, $\langle bd \rangle$, $\langle bdc \rangle$, $\langle bf \rangle$
$\langle c \rangle$	$\langle (ac)d(cf) \rangle$, $\langle (bc)(ae) \rangle$, $\langle b \rangle$, $\langle bc \rangle$	$\langle c \rangle$, $\langle ca \rangle$, $\langle cb \rangle$, $\langle cc \rangle$
$\langle d \rangle$	$\langle (cf) \rangle$, $\langle c(bc)(ae) \rangle$, $\langle (\_f)cb \rangle$	$\langle d \rangle$, $\langle db \rangle$, $\langle dc \rangle$, $\langle dcb \rangle$
$\langle e \rangle$	$\langle (\_f)(ab)(df)cb \rangle$, $\langle (af)cbc \rangle$	$\langle e \rangle$, $\langle ea \rangle$, $\langle eab \rangle$, $\langle eac \rangle$, $\langle eacb \rangle$, $\langle eb \rangle$, $\langle ebc \rangle$, $\langle ec \rangle$, $\langle ecb \rangle$, $\langle ef \rangle$, $\langle efb \rangle$, $\langle efc \rangle$, $\langle efcb \rangle$.
$\langle f \rangle$	$\langle (ab)(df)cb \rangle$, $\langle cbc \rangle$	$\langle f \rangle$, $\langle fb \rangle$, $\langle fbc \rangle$, $\langle fc \rangle$, $\langle fcb \rangle$

- Firstly, PrefixSpan algorithm finds the sequential patterns having a prefix $\langle a \rangle$. Moreover, in a sequence containing $\langle a \rangle$, only the subsequence prefixed with the first occurrence of $\langle a \rangle$ should be considered. For example, in a sequence $\langle (ef)(ab)(df)cb \rangle$, only the subsequence $\langle (\_b)(df)cb \rangle$ should be considered for mining sequential patterns having prefix $\langle a \rangle$.
- Then by scanning $\langle a \rangle$-projected database once, all the length of 2-elements sequential patterns having prefix $\langle a \rangle$ can be found. They are $\langle aa \rangle: 2, \langle ab \rangle: 4, \langle (ab) \rangle: 2, \langle ac \rangle: 4, \langle ad \rangle: 2, \langle af \rangle: 2$. Recursively, all sequential patterns having patterns the prefix can be partitioned into 6 subsets: (1) those having prefix $\langle aa \rangle$, (2) those having prefix $\langle ab \rangle$, …, and finally, (6) those having prefix $\langle af \rangle$. These subsets can be mined by constructing respective projected databases and mining each recursively as follows:
  - The $\langle aa \rangle$-projected database consists of only one non-empty (postfix) subsequences having prefix $\langle aa \rangle$: $\langle (\_b)(ac)d(cf) \rangle$ and since there is no hope to generate any frequent subsequence from a single sequence, the processing of $\langle aa \rangle$ projected database terminates.
  - The $\langle ab \rangle$-projected database consists of three postfix sequences: $\langle (\_c)(ac)d(cf) \rangle, \langle (\_c)a \rangle,$ and $\langle c \rangle$. Recursively mining $\langle ab \rangle$, the projected database returns four sequential patterns: $\langle (\_c) \rangle$, $\langle (\_c)a \rangle$, $\langle a \rangle$ and $\langle c \rangle$.

- $\langle ac \rangle, \langle ad \rangle, \langle af \rangle$ and other projected databases can be constructed and recursively mined similarly. The projected databases as well as the sequential patterns found are shown in Table 2.

No candidate sequence needs to be generated by PrefixSpan algorithm, which is a significant improvement compared to GSP and SPADE algorithms. Unlike Apriori-like algorithms, PrefixSpan algorithm only grows longer sequential patterns from the shorter frequent ones. PrefixSpan algorithm does not generate nor test any candidate sequence nonexistent in a projected database. PrefixSpan algorithm searches a much smaller space compared with GSP algorithm that generates and tests a substantial number of candidate sequences.

It is also important to note that in PrefixSpan algorithm a projected database is smaller than the original one, because only the postfix subsequences of a frequent prefix are projected into a projected database. In practice, the shrinking factors can be significant because (1) usually, only a small set of sequential patterns grow quite long in a sequence database, and thus the number of sequences in a projected database will become quite small when prefix grows; and (2) the projection only takes the postfix portion with respect to a prefix.

The major cost of PrefixSpan is the construction of projected databases i.e., forming projected databases recursively. In the worst case, PrefixSpan constructs a projected database for every sequential pattern. If there are a good number of sequential patterns, the cost is non-trivial. A bi-level projection scheme was proposed in PrefixSpan algorithm to reduce the number and the size of projected databases. Another approach – pseudo-projection technique was proposed to reduce the cost of projection substantially when a projected database can be held in the main memory. When the database can be held in main memory, instead of constructing a physical projection by collecting all the postfixes, one can use pointers referring to the sequences in the database as

a pseudo-projection. Every projection consists of two pieces of information: pointer to the sequence in database and offset of the postfix in the sequence. Pseudo-projection avoids physically copying postfixes. Thus, it is efficient in terms of both running time and space. However, it is not efficient if the pseudo-projection is used for disk-based accessing since random access disk space is very costly. Based on this observation, PrefixSpan algorithm always pursues pseudo-projection once the projected databases can be held in main memory. The extensive experimental results show that such an integrated solution, disk-based bi-level projection for disk-based processing and pseudo-projection when data can fit into the main memory, is always the clear winner in performance.

The performance comparison of GSP, SPADE and PrefixSpan algorithms shows that PrefixSpan algorithm has the best overall performance (Han et al., 2001) as displayed in the Figure 5. All the experiments were performed on a 233MHz Pentium PC machine with 128 megabytes main memory, running Windows. SPADE algorithm, although weaker than PrefixSpan algorithm in most cases, outperforms GSP algorithm as discussed in previous chapter. The algorithm performance studies also showed a general drawback applicable for all three previously discussed algorithms GSP, SPADE and PrefixSpan - when there are a large number of frequent subsequences, all three algorithms run slowly. The problem can be partially solved by closed sequential pattern mining, where closed subsequences are those sequential patterns containing no super-sequence with the same support.

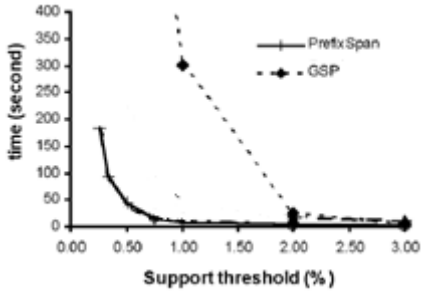


Figure 5. PrefixSpan and GSP algorithm comparison on different datasets.

2.5. Sequential Pattern Mining (SPAM) algorithm

SPAM algorithm (Sequential PAttern Mining) (Ayres et al., 2002) was proposed by J. Ayres et al. in 2002 for mining sequential patterns. The algorithm uses the first depth-first search strategy for mining sequential patterns and a vertical bitmap representation of the database that allows an efficient support counting.

SPAM algorithm assumes that the entire database and all data structures used by the algorithm completely fit into main memory and the authors of SPAM algorithm consider that in practise with the size of current main memories reaching gigabytes this assumption is not very limiting.

SPAM algorithm is based on a lexicographic tree of sequences. A similar approach has been used for the problem of mining frequent itemsets in MaxMiner (Bayardo, 1998) and MAFIA (Burdick, 2001).

SPAM algorithm considers that all sequences are arranged in a sequence tree:

- The root of the tree is labelled with θ;
- Recursively, if n is a node in the tree, then the children of n node are all nodes n' such that: $n \leq n'$ and $\forall m \in T: n' \leq m \rightarrow n \leq m$;
- The tree is finite due to the finite database instance as the input to SPAM algorithm;

- Each sequence in the tree can be considered as either a sequence-extended sequence (a sequence generated by adding a new transaction consisting of a single item to the end of its parent's sequence) or an itemset-extended sequence (a sequence generated by adding an item to the last itemset in the parent's sequence such that the item is greater than any item in that last itemset). For example, if we have a sequence $s = (\{a, b, c\}, \{a, b\})$, then $(\{a, b, c\}, \{a, b\}, \{a\})$ is a sequence-extended sequence of *s* and $(\{a, b, c\}, \{a, b, d\})$ is an itemset-extended sequence of *s*. Figure 6 shows a sample of a complete sequence tree for two items, a and b, given that the maximum size of a sequence is three. The top element in the tree is the null sequence and each lower level k contains all of the k-sequences, which are ordered lexicographically with sequence-extended sequences ordered before itemset-extended sequences. Each element in the tree is generated only by either an S-step or an I-step, e.g. sequence $(\{a, b\}, \{b\})$ is generated from sequence $(\{a, b\})$ and not from sequence $(\{a\}, \{b\})$ or$(\{b\}, \{b\})$.

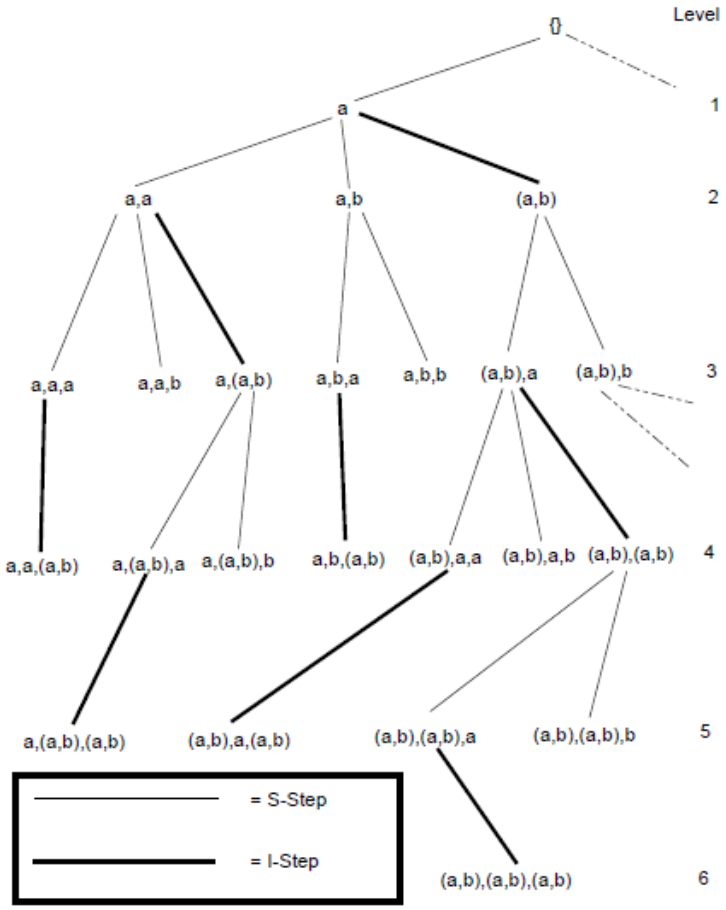


Figure 6. The lexicographic sequence tree in SPAM algorithm for a and b (max size of sequence is 3)

SPAM traverses the sequence tree described above in a standard depth-first manner. At each node n, the support of each sequence-extended child and each itemset-extended child is tested. If the support of a generated sequence s is greater than or equal to the user defined minimum support ε, the algorithm stores that sequence and repeats the DFS recursively on *s*. If the support of *s* is less than ε, then the algorithm does not repeat DFS on *s* by the Apriori principle, since any child sequence generated from s will not be frequent.

SPAM algorithm also uses the Apriori-based pruning technique to prune candidate s-extensions and i-extensions of a node n in the tree. For example, consider for the node $(\{a\})$ the possible itemset-extended sequences are

$(\{a, b\})$, $(\{a, c\})$ and $(\{a, d\})$. If $(\{a, c\})$ is not frequent, then $(\{a, b, c\})$ must also not be frequent by the Apriori principle.

For efficient counting SPAM algorithm uses a vertical bitmap representation of the data. A vertical bitmap is created for each item in the dataset, and each bitmap has a bit corresponding to each transaction in the dataset. If item *i* appears in transaction *j*, then the bit corresponding to transaction *j* of the bitmap for item *i* is set to one; otherwise, the bit is set to zero.

To enable efficient counting and candidate generation, the bitmap is partitioned so that all of the transactions of each sequence in the database will appear together in the bitmap (Figure 7).

CID	TID	$\{a\}$	$\{b\}$	$\{c\}$	$\{d\}$
1	1	1	1	0	1
1	3	0	1	1	1
1	6	0	1	1	1
-	-	0	0	0	0
2	2	0	1	0	0
2	4	1	1	1	0
-	-	0	0	0	0
-	-	0	0	0	0
3	5	1	1	0	0
3	7	0	1	1	1
-	-	0	0	0	0
-	-	0	0	0	0

Figure 7. Bitmap Representation of the dataset in Figure 6

If transaction m is before transaction n in a sequence, then the index of the bit corresponding to m is smaller than that of the bit corresponding to n. This bitmap idea extends naturally to itemsets. Efficient counting of support is one of the main advantages of the vertical bitmap representation of the data. To achieve this, the customer sequences are partitioned into different sets based on their lengths. If the size of a sequence is between $2^k + 1$ and 2^{k+1}, it is considered as a 2^{k+1}-bit sequence. The minimum value of k is 1 (i.e. any sequence of size smaller than or equal to 4 is considered as a 4-bit sequence).

Any sequence of size larger than 64 is ignored because in most practical databases this situation will not arise. Support counting for each customer then becomes a simple check as to whether the corresponding bitmap partition contains all zeros or not.

When comparing all three algorithms SPAM with SPADE and PrefixSpan on small datasets, the results show that SPAM outperforms SPADE by about a factor of 2.5 and better than an order of magnitude for reasonably large datasets. PrefixSpan outperforms SPAM slightly on very small datasets, but on large datasets SPAM outperforms PrefixSpan by over an order of magnitude. The primary reason that SPAM performs so well for large datasets is due to the bitmap representation of the data for efficient counting. The counting process is critical because it is performed many times at each recursive step, and SPAM handles it in an extremely efficient manner. For short datasets, the initial overhead needed to set up and use the bitmap representation in some cases outweighs the benefits of faster counting, and because of this PrefixSpan runs slightly faster for small datasets.

SPAM algorithm showed good results compared to SPADE and PrefixSpan, however it has a significant drawback, SPAM algorithm assumes that the entire database (and all data structures used for the algorithm) completely fit into main memory and therefore the algorithm cannot be applied to frequent sequence mining in big data or data streams.

2.6. Prime-Encoding Based Sequence Mining (PRISM) algorithm

PRISM (Prime-Encoding Based Sequence Mining) algorithm was proposed by K. Gouda et al. in 2009. This algorithm utilizes a vertical approach for enumeration and support counting, based on the novel notion of primal block encoding, which in turn is based on prime factorization theory.

There are three key aspects of PRISM algorithm:

- **The search space traversal strategy.** The sequence search tree can be defined recursively as follows: the root of the tree is at level zero and is labeled with the null sequence ∅. A node labeled with sequence S at level k, i.e., a k-sequence, is repeatedly extended by adding one item from I to generate a child node at the next level (k + 1), i.e., a (k + 1)-sequence. There are two ways to extend a sequence by an item: sequence extension (the item is appended to the sequential pattern as a new itemset) and itemset extension (the item is added to the last itemset in the pattern, provided that the item is lexicographically greater than all items in the last itemset). Thus, a sequence-extension always increases the size of the sequence, whereas, an itemset-extension does not. For example, if there is a node $S = ab \rightarrow a$ and an item b for extending S, then $ab \rightarrow a \rightarrow b$ is a sequence-extension, and $ab \rightarrow ab$ is an itemset extension. Figure 8 shows an example of a partial sequence search tree for two items, a and b, showing sequences up to size three; only the subtree under a is shown.

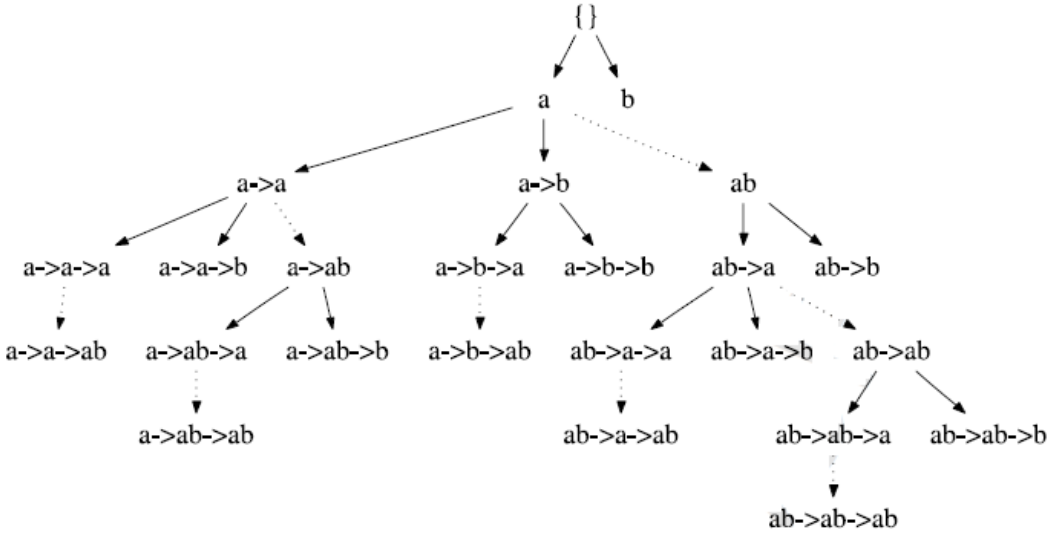


Figure 8. Partial Sequence Search Tree in PRISM: solid lines denote sequence extensions, whereas dotted lines denote itemset extensions

- The data structures used to represent the database and intermediate candidate information. Consider the example sequence database shown in Figure 9 (a), consisting of 5 sequences of different lengths over the items $\{a, b, c\}$. Let $G = \{2, 3, 5, 7\}$ be the base square-free prime generator set. For constructing the primal block encoding for a single item a, PRISM algorithm constructs the primal encoding of the positions within each sequence, e.g. since a occurs in positions 1, 4, and 6 (assuming indexes starting at 1) in sequence 1, the algorithm obtains the bit-encoding of a's occurrences: 100101. PRISM algorithm next pads this bit-vector so that it is a multiple of |G| = 4, to obtain A = 100101**00** (note: bold bits denote padding). Next PRISM algorithm computes $\nu(B, G) = \{\nu(B_i, G): 1 \leq i \leq m\}$, as the primal block encoding of B with respect to the base prime set G, so for a PRISM algorithm computes $\nu(A) = \nu(1001)\nu(0100) = \{14, 3\}$. The position encoding for a over all the sequences is shown in Figure 9 (b). PRISM algorithm next computes the primal encoding for the sequence ids. Since a occurs in all sequences, except for 4, a's sequence can be represented as a bit-vector A = 11101**000** after padding. This yields the primal encoding shown in Figure 9 (c), since $\nu(A) = \nu(1110)\nu(1000) = \{30, 2\}$. The full primal encoding for item a consists of all the sequence and position blocks, as shown in Fig. 3(d). A block $A_i = 0000 = 0$, with $\nu(A_i) = \{1\} = 1$, is also called an empty block. PRISM retains only the non-empty blocks in the primal encoding and eliminates the empty blocks. Fig. 3(e) shows the actual (compact) primal block encoding for item a. The first sequence block is 30, with factor-cardinality G = 3, which means that there are 3 valid (i.e., with non-empty position blocks) sequences in this block and for each of these, the offsets are stored into the position blocks. The benefit of this sparse representation becomes clear when we consider the primal encoding for c. Its full

encoding (see Figure 9 (d)) contains a lot of redundant information, which has been eliminated in the compact primal block encoding (see Figure 9 (e)).

sid	database sequence
1	$ab \rightarrow b \rightarrow b \rightarrow ab \rightarrow b \rightarrow a$
2	$ab \rightarrow b \rightarrow b$
3	$b \rightarrow ab$
4	$b \rightarrow b \rightarrow b$
5	$ab \rightarrow ab \rightarrow ab \rightarrow a \rightarrow bc$

(a)

sid	Bit-encoded pos	Primal-encoded pos
1	1001,0**100**	{14, 3}
2	100**0**	{2}
3	0**100**	{3}
4	000**0**	{1}
5	1111,0**000**	{210,1}

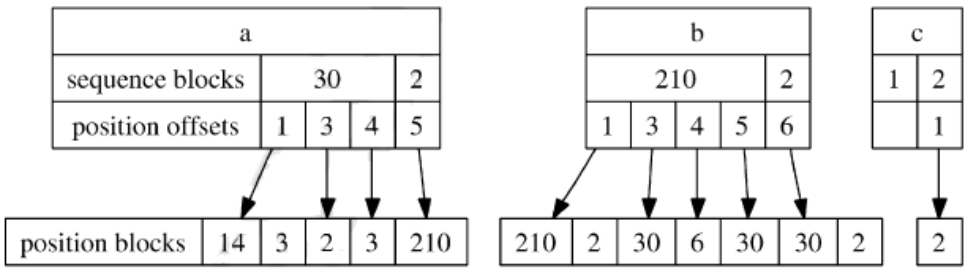
(b)

Bit-encoded sid	Primal-encoded sid
1110,1**000**	{30,2}

(c)

Item	Sequence Blocks	Position Blocks
a	{30,2}	{14, 3}, {2}, {3}, {1}, {210,1}
b	{210,2}	{210, 2}, {30}, {6}, {30}, {30,2}
c	{1,2}	{1, 1}, {1}, {1}, {1}, {1,2}

(d)



(e)

Figure 9. The example of primal block encoding in PRISM algorithm: (a) Example database. (b) Position encoding for a. (c) Sequence

encoding for a. (d) Full primal blocks for a, b and c. (e) Primal block encoding for a, b and c

- Counting the support for candidates for itemset and sequence extensions can be determined via primal block joins. Assume some node S in the sequence search tree, where S is a k-sequence. Let P be the parent node of S, which means that P is a (k −1) length prefix of S. As illustrated in Figure 8, S is extended by considering the other sibling nodes of S under parent P. For example, if P = ab, and S = ab→a, then other extensions of S can be generated by also considering which extensions of P (the siblings of S) are frequent. In Figure 8, the only other sibling is T = ab →b. This suggests that both items a and b can be used to extend S = ab →a. There are two possible sequence extensions, namely ab →a→a and ab→a→b, and one itemset extension ab→ab. In fact, the support for any extension is always the result of joining the primal block encodings of two (possible the same) sibling nodes under the same prefix P. The frequent sequence enumeration process starts with the root of the search tree as the prefix node P =∅, and PRISM algorithm assumes that initially we know the primal block encodings for all single items. PRISM algorithm then recursively extends each node in the search tree, computes the support via the primal block joins, and retains new candidates (or extensions) only if they are frequent. The search is essentially depth-first, the main difference being that for any node S, all of its extensions are evaluated before the depth-first recursive call. When there are no new frequent extensions found, the search stops. Detailed primal block join operations for primal block itemset and sequence joins using the primal encodings for items a and b are shown in Figure 9 (e).

Extensive evaluations on both synthetic and real datasets show that the PRISM algorithm outperforms popular sequence mining methods like SPADE, PrefixSpan and SPAM, by an order of magnitude or more. Based on the extensive set of results on diverse datasets, PRISM is the fastest algorithm, and runs for lower support values than competing methods. SPAM generally works only for smaller datasets due to its very high memory consumption; when it runs, SPAM is generally the second best. SPADE and PrefixSpan do not suffer from the same problems as SPAM, but they are much slower than Prism, or they fail to run for lower support values, when the database parameters are large. LAPIN (Yang et al., 2006) has also been shown to outperform existing approaches, especially on long, dense datasets.

2.7. Last Position Induction (LAPIN) algorithm

There has been a great deal of effort put into frequent sequence mining in recent years and number of algorithms proposed. However the performance of these frequent sequence mining algorithms is still far from satisfactory because of two main challenges: large search spaces and the ineffectiveness in handling dense datasets. To offer a solution to the above challenges, Z. Yang et al. in 2006 proposed Last Position Induction algorithm (LAPIN) for sequential pattern mining, which is based on the simple idea that the last position of an item, α, is the key to judging whether or not a frequent k-length sequential pattern can be extended to be a frequent (k+1)-length pattern by appending the item α to it. LAPIN can largely reduce the search space during the mining process, and is very effective in mining dense datasets.

According to LAPIN algorithm for any sequence database, the last position of an item is the key used to judge whether or not the item can be appended to a given prefix (k-length) sequence.

Example 1. The sequence database S shown in Figure 10 (a) will be used in this running example with min_support $= 2$. When scanning the database in Figure 10 (a) for SID Last Position of SE Item the first time, Figure 10 (b) is obtained, which is a list of the last positions of the 1-length frequent sequences in ascending order.

SID	Sequence
10	a c (b c) d (a b c) a d
20	b (c d) a c (b d)
30	d (b c) (a c) (c d)

(a) Sequence DB

SID	Last Position of SE Item
10	b_{last}=5 c_{last}=5 a_{last}=6 d_{last}=7
20	a_{last}=3 c_{last}=4 b_{last}=5 d_{last}=5
30	b_{last}=2 a_{last}=3 c_{last}=4 d_{last}=4

(b) Last positions of items

Figure 10. The sample database and last position of items

Suppose that there exists a prefix frequent sequence $\langle a \rangle$, and its positions in Figure 10 (a) are 10:1, 20:3, 30:3, where sid:eid represents the sequence ID and the element ID. Then, Figure 10 (b) is checked to obtain the first indices whose positions are larger than $\langle a \rangle$'s, resulting in 10:1, 20:2, 30:3, i.e., ($10{:}\,\mathrm{b}_{\mathrm{last}} = 5$, $20{:}\,\mathrm{c}_{\mathrm{last}} = 4$, and $30{:}\,\mathrm{c}_{\mathrm{last}} = 4$), symbolized as "↓". Starting from these indices to the end of each sequence, the support of each passed item is incremented, resulting in $\langle a \rangle$: 1; $\langle b \rangle$: 2; $\langle c \rangle$: 3; and $\langle d \rangle$: 3, from which it is determined that $\langle ab \rangle$, $\langle ac \rangle$ and $\langle ad \rangle$ are the frequent patterns.

Let D be the average number of customers (i.e., sequences) in the projected DB, L be the average sequence length in the projected DB, N be the average total number of the distinct items in the projected DB, and m be the distinct item recurrence rate or density in the projected DB. Then $m = \frac{L}{N}$ ($m \geq 1$), and the relationship between the runtime of PrefixSpan (T_{ps}) and the runtime of LAPIN (T_{lapin}) in the support counting part is

$$\frac{T_{ps}}{T_{lapin}} = \frac{D \times L}{D \times N} = m$$

Because support counting is usually the most costly step in the entire mining process, the equality above illustrates the main reason why LAPIN is faster than PrefixSpan for dense datasets, whose m (density) can be very high.

The performance study on synthetic and real life datasets demonstrates that LAPIN outperforms PrefixSpan (Pei et al., 2004) by up to an order of magnitude on long pattern dense datasets. LAPIN algorithm was first compared to PrefixSpan using several small-, medium-, and large- sized datasets. The results clearly illustrate that PrefixSpan is slower than LAPIN for both the medium and large datasets, because the searched spaces of the two datasets in PrefixSpan were much larger than that in LAPIN. For the small dataset, the initial overhead needed to set up meant that LAPIN was slower than PrefixSpan.

2.8. Conclusions

Since 1995 there were a number of algorithms proposed for frequent sequence mining starting from GSP algorithm by R. Agrawal and R. Srikant that works according to the Apriori-principle. GSP became a state-of-the-art algorithm in the frequent sequence mining. Other exact state-of-the-art frequent sequence mining algorithms are as follows: SPADE algorithm in 2001 by M. J. Zaki et al., PrefixSpan algorithm in 2001 J. Han et al., SPAM algorithm in 2002 by J. Ayres et al., LAPIN algorithm in 2006 by Z. Yang et al. and PRISM algorithm in 2007 by K. Gouda et al.

PrefixSpan shows overall best performance, then SPADE and SPAM outperforms GSP, which is in the last place. PrefixSpan outperforms SPAM slightly on very small datasets, but on large datasets SPAM outperforms PrefixSpan and SPADE by over an order of magnitude. However SPAM assumes that the entire database (and all data structures used for the algorithm)

completely fit into main memory and therefore the algorithm could not be applied to frequent sequence mining in big data or data streams.

Based on comparisons of the algorithms' performance, PRISM and LAPIN algorithms outperform SPADE, PrefixSpan and SPAM algorithms by at least an order of magnitude. There have been no direct algorithm performance studies between PRISM and LAPIN algorithms to identify which of these is the most outstanding algorithm; both algorithms have been compared to other frequent sequence mining algorithms separately.

All mentioned algorithms for frequent sequence mining are exact algorithms and require reading the entire original database at least several times.

Chapter 3: New approximate algorithms for mining frequent sequences

3.1. Introduction

There are a number of exact frequent sequence mining algorithms proposed since 1995 (GSP, SPADE, SPAM, PrefixSpan, LAPIN, PRISM, etc.) and a number of improvements were proposed that significantly improved the performance of exact frequent sequence mining algorithms. All exact algorithms for mining frequent sequences make multiple passes over the original database. Having in mind the challenges with increasing the volume and variety of big data, it is obvious that in many areas, where the computation speed is critical, only approximate algorithms for data analysis could deliver required quick results.

To our best knowledge all previously proposed approximate frequent sequence mining methods do not use a theoretical estimate of the probability of error made by the method when identifying frequent patterns in the original database. These methods provide only the empirical evidence based on extensive experiments and observations of algorithm results on different databases.

In the following chapters the detailed description is given about proposed novel approximate frequent sequence mining methods that have a theoretical estimate of the probability of error made by proposed methods when classifying sequences as frequent or rare:

- RSM (Random Sampling Method);
- MRM (Multiple Re-sampling Method);
- MPBM (Markov Property Based Method).

The results of above proposed approximate methods were published in (Pragarauskaitė, Dzemyda, 2009, 2010, 2011, 2013).

The performance of methods is compared to other approximate and exact frequent sequence mining methods and recommendations given on how to select parameters used in proposed methods to achieve more accurate results.

3.2. Overview of existing approximate algorithms for frequent sequence mining

In this chapter two existing approximate algorithms (ApproxMAP and ProMFS) are described. Both algorithms approximately determine frequent and rare sequences in the original database.

3.2.1. Approximate Multiple Alignment Pattern mining (ApproxMAP) algorithm

The ApproxMAP (APPROXimate Multiple Alignment Pattern mining) algorithm (Kum et al., 2003) was proposed by H. C. Kum et al. in 2003. The idea of ApproxMAP algorithm is that, instead of finding a huge set of exact patterns, the algorithm identifies patterns (called consensus patterns) approximately shared by many sequences. The goal of the algorithm is to organize and summarize a sequence of sets to uncover the underlying consensus patterns in the data. ApproxMAP algorithm uses clustering as a pre-processing step to group similar sequences, and then mines the underlying consensus patterns in each cluster directly through multiple alignments. A novel structure of weighted sequences is proposed to summarize and compress the alignment information. ApproxMAP algorithm was the first study on mining consensus patterns from sequence databases and is effective and efficient for mining long sequences and is robust to noise.

ApproxMAP algorithm uses the following definitions:

Assume $S_1 = \{X_1, \dots, X_n\}$ and $S_2 = \{Y_1, \dots, Y_m\}$ are sequential patterns, where $X_1, \dots, X_n$ and $Y_1, \dots, Y_m$ are all items in the set.

A hierarchical edit distance between S_1 and S_2 is defined as follows:

$$D(i,j) = min \begin{cases} D(i-1,j) + INDEL(X_i) \\ D(i,j-1) + INDEL(Y_j) \\ D(i-1,j-1) + REPL(X_i, Y_j) \end{cases},$$

where $1 \le i \le n$, $1 \le j \le m$, INDEL() denotes an insertion or deletion operation and REPL() denotes a replacement operation.

A normalized hierarchical edit distance is denoted as follows:

$$dist(S_1, S_2) = \frac{D(n,m)}{max\{\|S_1\|, \|S_2\|\}}$$

Using the normalized hierarchical edit distance denoted above, a density-based clustering algorithm can be applied to cluster sequences. A sequence is called "dense" if there are many sequences similar to it in the database. In particular, for each sequence S_i in a database S, let $d_1, \dots, d_k$ be the k (user-specificied) smallest non-zero values of $dist(S_i, S_j)$, where $S_j \neq S_i$, is a sequence in S. Then, the density is defined as follows:

$$Density(S_i) = \frac{n}{\|S\| d},$$

where $d = max\{d_1, \dots, d_k\}$ and $n = \left\|\{S_j \in S | dist(S_i, S_j) \le d\}\right\|$. n is the number of sequences in the k-nearest neighbour space.

ApproxMAP algorithm works in the following three steps:

1) *Clustering sequences in the original database based on the similarity*. The clustering algorithm takes as input a set of sequences $S = \{S_i\}$ and the user defined number of nearest neighbours k that the algorithm will search (a larger k value tends to merge more sequences and results in a smaller

number of large clusters, while a smaller k value tends to break up clusters). The algorithm outputs a set of clusters $\{C_j\}$, where each cluster is a set of sequences. The clustering is done in the following three steps:

i) Initialize every sequence as a cluster. For each sequence S_i in cluster C_{S_i}, set $Density\left(C_{S_i}\right) = Density\ (S_i)$.

ii) Merge nearest neighbours based on the density of sequences. For each sequence S_i, let $S_{i_1}, \ldots, S_{i_n}$ be the nearest neighbor of S_i. For each $S_j \in \{S_{i_1}, \ldots, S_{i_n}\}$, merge cluster C_{S_i} containing S_i with a cluster C_{S_j} containing S_j , if $Density(S_i) < Density(S_j)$ and there exists no S'_j having $dist(S_i, S'_j) < dist(S_i, S_j)$ and $Density(S_i) < Density(S'_j)$. Set the density of the new cluster to $max\{Density(C_{S_i}), Density(C_{S_j})\}$.

iii) Merge based on the density of clusters. For all sequences S_i such that S_i has no nearest neighbour with density greater than that of S_i, but has some nearest neighbour, S_j , with density equal to that of S_i, merge the two clusters C_{S_j} and C_{S_i} containing each sequence if $Density(CSj) > Density(CSi)$.

2) *Multiple Alignment and Pattern Generation***.** Once sequences are clustered, sequences within a cluster are similar to each other. Then the method for multiple alignments of sequences is performed to summarize the general pattern in each cluster and discover the trend. In the clustering step (1), the density for each sequence was calculated, so only sorting is needed for all sequences within a cluster according to the density in descending order. To store the alignment results effectively, ApproxMAP proposes a notion of weighted sequence $WS = \langle X_1 : v_1, \ldots, X_l : v_l \rangle : n$, that carries the following information:

(1) the current alignment has n sequences; n is called the global weight of the weighted sequence;

(2) in the current alignment, v_i sequences have a non-empty itemset X_i aligned in the i^{th} itemset, where $(1 \leq i \leq l)$;

(3) an itemset in the alignment is in the form of $X_i = (x_{j1}: w_{j1}, \dots, x_{jm}: w_{jm})$.

3) *Generation of Consensus Patterns (longest approximate sequential patterns for each cluster).* In order to extract consensus patterns, a weighted sequence is derived for each cluster using multiple alignments to compress the sequential pattern information in the cluster. And then the longest consensus pattern best representing the cluster is generated from the weighted sequence. Intuitively, a consensus pattern is shared by many sequences and covers many short patterns. A weighted sequence records the statistics of the alignment of the sequences in a cluster. Intuitively, a pattern can be generated by picking up parts of a weighted sequence shared by most sequences in the cluster. For a weighted sequence

$$WS = \langle (x_{11}: w_{11}, \dots, x_{1m_1}: w_{1m_1}) : v_1, \dots, (x_{l1}: w_{l1}, \dots, x_{lm_1}: w_{lm_1}): v_l \rangle : n,$$

the strength of item $x_{ij} : w_{ij}$ in the i^{th} itemset is defined as $\frac{w_{ij}}{n} \cdot 100\%$. Clearly, an item with a larger strength value indicates that the item is shared by more sequences in the cluster. A user can specify a strength threshold $(0 \leq min\_strength \leq 1)$. A consensus pattern P can be extracted from a weighted sequence by removing items in the sequence whose strength values are lower than the threshold.

ApproxMAP algorithm distinguishes itself from the previous studies because it proposes the theme of approximate sequential pattern mining, which reduces number of patterns substantially and provides much more accurate and informative insights into sequential data.

The authors of ApproxMAP algorithm do not provide the performance study in terms of the computing speed, only the precision based on the empirical

experiment on a real data set of welfare services accumulated over a few years in North Carolina State. The services have been recorded monthly for children who had a substantiated report of abuse and neglect, and were placed in foster care. There were 992 such sequences and ApproxMAP algorithm found 15 interpretable and useful patterns and some of them required additional scan of the original database to do extensive checks. The experiment showed that ApproxMAP algorithm helps to find the insights in the original database, however it is reading the original database multiple times and focuses more on finding consensus patterns according to the parameters defined by the user, rather than the computation speed of the algorithm. It is also important to note that both user-specified parameters (the number of nearest neighbours k that the algorithm will search and consensus strength threshold) need extensive knowledge about the dataset in order to get useful outcome.

3.2.2. Probabilistic algorithm for Mining Frequent Sequences (ProMFS)

The approximate ProMFS algorithm (PRObabilistic algorithm for Mining Frequent Sequences) (Tumasonis, Dzemyda, 2004) was proposed by R. Tumasonis et al. in 2004, which is based on the statistical characteristics of the elements appearance in original database and their order. The algorithm builds a much shorter sample of the original database, analyses it using the exact GSP algorithm (Srikant, Agrawal, 1996) and makes assumptions on the frequent and rare sequences in the original database based on the results of analysis of the shorter sample. The algorithm is purely based on the empirical analysis and is not based on the theoretical proof using the statistical methods. The ProMFS algorithm performs the following steps:

1) Determining the following statistical characteristics of the position and interposition of elements in the original database during one full database scan:

i) The probability of element $i_j \in S, j = 1, \dots m$ occurrence in the original database S: $P(i_j) = \frac{V(i_j)}{N}$, where $V(i_j)$ is the number of such elements i_j in the original database S and N is the length of the original database;

ii) The probability $P(i\ j \mid iv)$ for the element i_v to appear after i_j in the original database S;

iii) The average distance D(i j | iv) between the elements i_j and i_v in the original database S;

iv) The matrix of average distances $A_{jv} = Average(D(i\ j \mid iv))$, where $j, v = 1, \dots, m$ in the original database S.

2) Generating a new much shorter sample (also called model sequence) $\bar{S}_n$ of the original database in the original database S according to the statistical characteristics determined in item 1). The first element in the sample $\bar{S}_n$ is i_t, that has maximum probability max($P(i_t)$), all other elements are constructed according to the numeric characteristic max $(Q(i_j, \tilde{s}_r)), i_j \in S, \tilde{s}_r \in \bar{S}_n, 1 \le r \le n$. First of all, the values of $Q(i_j, \tilde{s}_r)$ equals to 0, then additional function $\rho(\tilde{s}_r, a_{rj})$ is introduced that increments the values of characteristics $Q(i_j, \tilde{s}_r)$ by 1: $\rho(\tilde{s}_r, a_{rj}) \rightarrow Q(i_j, 1 + \tilde{a}_r) = Q(i_j, \tilde{a}_r) + 1, j = 1, \dots, m$.

3) Analyzing the new model sequence $\bar{S}_n$ with the exact GSP algorithm and comparing the results of true frequencies of sequences in the original database with the empirical frequencies identified by the GSP algorithm on the sample.

The following example illustrates how the ProMFS algorithm works. Consider the original database $S = ABCCCBBCABCABCABCBABCCCABCAABABCABC$ and the length of the original database S is $N = 35$. The sequence is considered frequent, if it occurs in the original database S not less than 4 times (the minimum support is equal to 4). The size of model sequence (sample) $\bar{S}_n$ is equal to $n = 8$.

After a full scan of the original database S, the following statistical characteristics are calculated:

$$P(A) = \frac{10}{35} \approx 0.2857; P(B) = \frac{12}{35} \approx 0.3429; P(C) = \frac{13}{35} \approx 0.3714;$$

$$P(A \mid A) = 0.1; P(A \mid B) = 0.9; P(A \mid C) = 0;$$

$$P(B \mid A) \approx 0.1667; P(B \mid B) = 0.0833; P(B \mid C) \approx 0.750;$$

$$P(C \mid A) \approx 0.4615; P(C \mid B) = 0.1538; P(C \mid C) \approx 0.3077.$$

The average distances matrix A is displayed in Table 3.

Table 3. The matrix A of average distances in ProMFS algorithm

	A	*B*	*C*
A	3.58	1.10	2.50
B	2.64	2.91	1.42
C	2.33	2.25	2.67

The sample $\bar{S}_n$ of the original database S is formed in the following way:

- First of all, the model sequence (sample) $\bar{S}_n$ is empty:

r	1	2	3	4	5	6	7	8
A	0	0	0	0	0	0	0	0
B	0	0	0	0	0	0	0	0
C	0	0	0	0	0	0	0	0
Model sequence	-	-	-	-	-	-	-	-

- The first element in $\bar{S}_n$ is chosen according to the maximum probability $P(i_j)$ and this example it is C. Then $Q(i_j, \tilde{s}_1), j = 1,2,3$ is recalculated according to the average distances:

r	1	2	3	4	5	6	7	8
A	0	0	**1**	0	0	0	0	0
B	0	0	**1**	0	0	0	0	0
C	0	0	0	**1**	0	0	0	0
Model sequence	***C***							

- All three values $Q(i_j, \tilde{s}_1), j = 1,2,3$ are equal. Moreover, they are equal to zero. Therefore, $\tilde{s}_2$ will be determined by maximal value of conditional probabilities max(P(C|A), P(C|B), P(C|C)=P(C|A)=

0.4615. Therefore, $\tilde{s}_2 = A$. Then recalculate $Q(i_j, \tilde{s}_2), j = 1,2,3$ according to the average distances. The situation becomes as follows:

r	1	2	3	4	5	6	7	8
A	0	0	1	0	0	**1**	0	0
B	0	0	**2**	0	0	0	0	0
C	0	0	0	1	**1**	0	0	0
Model sequence	C	A						

- The next three steps are performed in the similar way and the situation becomes as follows:

r	1	2	3	4	5	6	7	8
A	0	0	1	0	0	3	**1**	0
B	0	0	2	0	0	2	**1**	0
C	0	0	0	1	2	0	1	**1**
Model sequence	C	A	B	C	C			

- Following the same approach, the model sequence $\bar{S}_n = \text{CABCCABC}$ is constructed.

Then the model sequence $\bar{S}_n$ is analyzed using the exact GSP algorithm. In this particular example GSP algorithm determined that the longest frequent sequence of sample $\bar{S}_n$ is ABC when the minimum support is set to 2. Moreover, the second frequent sequence is CAB for the same minimum support. The true frequences in the original database for sequence ABC is 8 and for CAB is 5, so the ProMFS algorithm correctly classified these sequences as frequent. However, the subsequence BCA, whose true frequency is 5 in the original database, has not been determined by the analysis of the sample $\bar{S}_n$.

The performance study of ProMFS algorithm was performed on the synthetic database of 100000 elements and the analyzed sample consisted of 40 elements. The study showed that ProMFS algorithm allows to save the computing time in a large extent compared to the exact algorithms because instead of analyzing the original database, it analyzes newly constructed sample of the original database (see Figure 11).

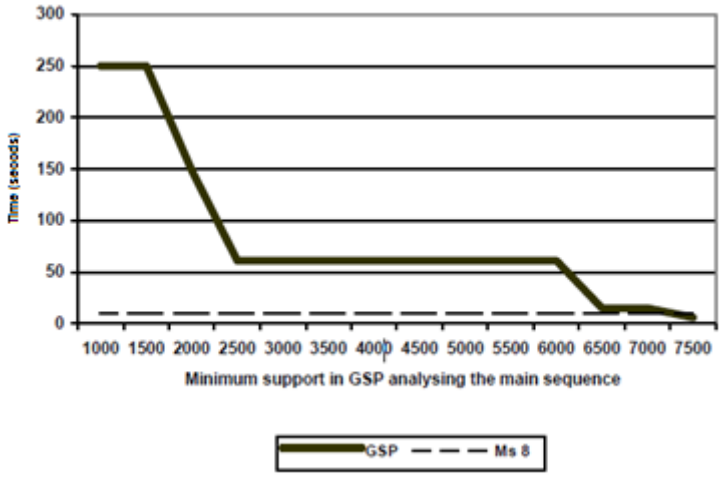


Figure 11. Computing time used by GSP and ProMFS (minimum support is 8)

However, ProMFS algorithm still requires reading the entire original database once in order to calculate statistical characteristics required for the construction of sample.

The precision of the algorithm and the number of identified frequent sequences vary up to 30% in the performance study and the results are dependant on the data (see Figure 12).

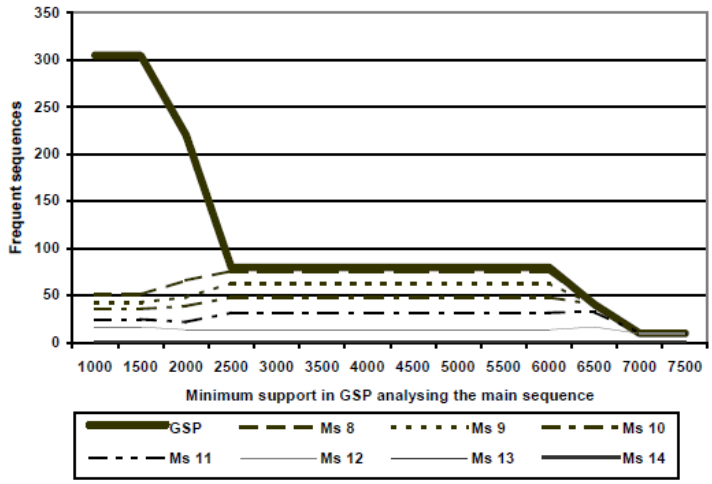


Figure 12. The number of frequent sequences found by GSP and ProMFS (minimum support = 8,..,14)

The drawbacks of the ProMFS algorithm are the fact that it requires reading the entire original database at least once and it also does not have any

theoretical estimation, therefore require extensive empirical tests before it could be used in real-world applications.

3.3. Novel approximate method: Random Sampling Method (RSM)

The random sampling method (RSM) analyzes the random sample of the original database and makes statistical inferences about the original database with estimated error probability. Let the original database be a set $S = \{x_1, x_2, \dots, x_N\}$ whose elements x_i can take M different values in a set $V = \{\alpha_1, \alpha_2, \dots, \alpha_M\}$. A sequence $(a_{i_1}, \dots, a_{i_m})$ is called frequent if (# is a number):

$$\frac{1}{N-m+1} \#\left\{j \in \{1, \dots, N-m+1\}: x_j = \alpha_{i_1}, x_{j+1} = \alpha_{i_2}, \dots, x_{j+m-1} = \alpha_{i_m}\right\} \geq \varepsilon,$$

where $\varepsilon \in (0,1)$ is a given threshold also known as a minimum support.

The description of RSM method was published in (Pragarauskaitė, Dzemyda, 2011). Detailed description of the creation of the random sample of the original database and the RSM method is explained in the following chapters.

3.3.1. Creating a random sample from the original database

The original database random sample $\bar{S}_n$ is chosen as follows:

1. A random sample $\eta_1, \eta_2, \dots, \eta_n$ of a random variable η taking values $1, 2, \dots, N$ each with probability $\frac{1}{N}$, is generated.
2. When searching for the first level (one-element) frequent sequences, the random sample $\bar{S}_n$ for elements a_i is $x_{\eta_1}, x_{\eta_2}, \dots, x_{\eta_n}$. The second level random sample for element pairs $a_i a_j$ is $(x_{\eta_1}, x_{\eta_1+1})$, $(x_{\eta_2}, x_{\eta_2+1})$, …, $(x_{\eta_n}, x_{\eta_n+1})$. The *k*-th

level random sample for elements $a_i \dots a_k$ is $(x_{\eta_1}, \dots, x_{\eta_1+k-1})$, $(x_{\eta_2}, \dots, x_{\eta_2+k-1})$,…,$(x_{\eta_n}, \dots, x_{\eta_n+k-1})$, etc. If the sample is formed with replacement then some integers η_i can be the same. The random sample formed without replacement consists of non-repeatable numbers η_i, by eliminating all repeatable numbers and additionally generating new numbers until we get a sequence $\eta_1, \eta_2, \dots, \eta_n$ without repeatable integers.

3.3.2. Selecting the exact algorithm for analyzing the random sample

According to the RSM method, once the random sample $\bar{S}_n$ of the original database is created, then it could be analyzed using any exact algorithm in order to get a precise classification of frequent and rare sequences. Based on this classification, the assumptions about frequent sequences in the original database are made and the classification errors made by the approximate algorithm are estimated.

In this thesis the GSP algorithm is used to get the precise classification of frequent and rare sequences in the random sample, because other approximate algorithm that will be used for performance comparison with RSM also uses GSP to analyze the sample of the original database.

RSM method does not limit the selection of exact algorithm for the random sample analysis and any other exact algorithm could be selected according to the algorithm performance studies provided in the Chapter 2. The best results in terms of the computing speed are demonstrated by PRISM or LAPIN algorithms as shown in the performance study in the Chapter 2.

3.3.3. Analyzing the sample using Random Sampling Method (RSM)

Using exact frequent sequence algorithm GSP, the empirical frequencies of sequences $a_{i_1}, a_{i_2}, \dots, a_{i_k}$ in the random sample $\bar{S}_n$ are determined:

$$\bar{p}_n(a_{i_1}, \dots, a_{i_k}) = \frac{\#\{j: S_{\eta_j}=a_{i_1}, S_{\eta_j+1}=a_{i_2}, \dots, S_{\eta_j+k-1}=a_{i_k}\}}{n}, k = 1,2, \dots.$$

Then we choose a number $\delta > 0$ such that $(0 < \varepsilon - \delta < \varepsilon + \delta < 1)$. Sequences $a_{i_1}, a_{i_2}, \dots, a_{i_k}$ are classified into the following 3 classes:

1. if $\bar{p}_n(a_{i_1}, \dots, a_{i_k}) \geq \varepsilon + \delta$, then the sequence $a_{i_1}, \dots, a_{i_k}$ is assigned to the class of frequent sequences;
2. if $\bar{p}_n(a_{i_1}, \dots, a_{i_k}) \leq \varepsilon - \delta$, then the sequence $a_{i_1}, \dots, a_{i_k}$ is assigned to the class of rare sequences;
3. if $\bar{p}_n(a_{i_1}, \dots, a_{i_k}) \in (\varepsilon - \delta, \varepsilon + \delta)$, then the sequence $a_{i_1}, \dots, a_{i_k}$ is assigned to the class of intermediate sequences.

The error probabilities can be evaluated as follows. Let's take any sequence $a_{i_1}, \dots, a_{i_k}$. There are two types of errors:

1. a sequence is assigned to the class of frequent sequences, but the sequence is rare;
2. a sequence is assigned to the class of rare sequences, but the sequence is frequent.

Let $p = p(a_{i_1}, \dots, a_{i_k})$ be the true but unknown frequency of the sequence $(a_{i_1}, \dots, a_{i_k})$ and the empirical frequency $\bar{p}_n = \bar{p}_n(a_{i_1}, \dots, a_{i_k})$. It is obvious, that the first type error probability is less than

$$\max_{p<\varepsilon} P(\overline{p_n} - p > \delta), \quad (1)$$

and the second type error probability is less than

$$\max_{p \geq \varepsilon} P(\overline{p_n} - p < -\delta).$$

(2)

When estimating these probabilities it is convenient to use the following scheme. Let‘s define random variables

$$Z_i = \begin{cases} 1, \text{ if } x_{\eta_i} = a_{i_1},\ x_{\eta_i+1} = a_{i_2},\ \dots,\ x_{\eta_i+k-1} = a_{i_k} \\ 0, \quad \text{otherwise} \end{cases}, \quad i = 1, \dots, n.$$

By the construction of the sequence $\eta_1, \eta_2, \dots, \eta_n$, the random variables $Z_1, Z_2, \dots, Z_n$ are independent and identically distributed with the mean $EZ_i = p$, and the variance $DZ_i = p(1-p)$.

Probabilities (1) and (2) can be estimated using standard statistical methods: using the properties of binomial distribution for a sample with replacement and hypergeometric distribution for a sample without replacement.

Let‘s define the random variable

$$\Sigma_n = Z_1 + Z_2 + \cdots + Z_n.$$

For the sample without replacement, the distribution function of the statistics Σ_n is given by

$$F(l, M) = \sum_{i=0}^{l} \frac{C_M^i C_{N-M}^{n-i}}{C_N^n}, l = 0,1, \dots, M,$$

where $M = \#\{j: x_j = a_{i_1}, x_{j+1} = a_{i_2}, \dots, x_{j+k-1} = a_{i_k}\}$. Let us fix $\alpha \epsilon (0,1)$ and define the numbers $\underline{M}$ and $\overline{M}$ as follows:

- $\underline{M}$ is the minimum integer such that

$$F(\Sigma_n - 1, \underline{M}) \geq 1 - \alpha,$$

- $\overline{M}$ is the maximum integer such that

$$F(\Sigma_n - 1, \overline{M}) \leq \alpha.$$

The integers $\underline{M}$ and $\overline{M}$ are the parameter's M lower and upper $(1-\alpha)$ confidence bounds. Hence,

$$P\{M \geq \underline{M}\} = P\left\{p(a_{i_1}, \dots, a_{i_k}) \geq \frac{\underline{M}}{N}\right\} \geq 1 - \alpha,$$

$$P\{M \leq \overline{M}\} = P\left\{p(a_{i_1}, \dots, a_{i_k}) \leq \frac{\overline{M}}{N}\right\} \geq 1 - \alpha.$$

LEMMA 1. If the sample size n is sufficiently large, then the asymptotic estimates of error probabilities are effective. By the central limit theorem, for all $a \leq b$ we get:

$$P(a\frac{\sqrt{p(1-p)}}{\sqrt{n}} \leq \overline{p_n} - p \leq b\frac{\sqrt{p(1-p)}}{\sqrt{n}}) \to \Phi(b) - \Phi(a), n \to \infty,$$

where Φ is the standard normal distribution function.

Proof. Central limit theorem claims that for all $a \leq b$:

$$P(a \leq \frac{Z_n - EZ_n}{\sqrt{DZ_n}} \leq b) \to \Phi(b) - \Phi(a), n \to \infty$$

In our case $Z_n = n\bar{p}_n$ therefore we get

$$\frac{Z_n - EZ_n}{\sqrt{DZ_n}} = \frac{n\bar{p}_n - np}{\sqrt{np(1-p)}} = \sqrt{\frac{n}{p(1-p)}}(\bar{p}_n - p)$$

We rewrite Central limit theorem as follows

$$P(a\frac{\sqrt{p(1-p)}}{\sqrt{n}} \leq \overline{p_n} - p \leq b\frac{\sqrt{p(1-p)}}{\sqrt{n}}) \to \Phi(b) - \Phi(a), n \to \infty$$

Lemma is proved.

If $a = -\infty$, then for each b

$$P(\overline{p_n} - p \leq b\frac{\sqrt{p(1-p)}}{\sqrt{n}}) \to \Phi(b), n \to \infty. \quad (3)$$

If $b = +\infty$, then for each a

$$P\left(a\frac{\sqrt{p(1-p)}}{\sqrt{n}} \leq \overline{p_n} - p\right) \to 1 - \Phi(a), n \to \infty. \quad (4)$$

If n is sufficiently large, then using (3) and (4), we have

$$\max_{p<\varepsilon} P(\overline{p_n} - p > \delta) \approx \max_{p<\varepsilon}\left(1 - \Phi(\delta\frac{\sqrt{n}}{\sqrt{p(1-p)}})\right) \leq 1 - \Phi(\delta\frac{\sqrt{n}}{\sqrt{\varepsilon_0(1-\varepsilon_0)}}), \quad (5)$$

where $\varepsilon_0 = \min\left(\varepsilon, \frac{1}{2}\right)$, and

$$\max_{p\geq\varepsilon} P(\overline{p_n} - p < -\delta) \approx \max_{p\geq\varepsilon}\left(\Phi(-\delta\frac{\sqrt{n}}{\sqrt{p(1-p)}})\right) \leq \Phi(-\delta\frac{\sqrt{n}}{\sqrt{\varepsilon_1(1-\varepsilon_1)}}), \quad (6)$$

where $\varepsilon_1 = \max\left(\frac{1}{2}, \varepsilon\right)$.

If $\bar{p}_n \in (\varepsilon - \delta, \varepsilon + \delta)$, then the classification is undefined, because the classification error probability can be large. The classification error probability depends on how much the true frequency p differs from ε. Assume that $p = \varepsilon$. According to the central limit theorem $P(\bar{p}_n \geq \varepsilon) \to \frac{1}{2}, n \to \infty$ and $P(\bar{p}_n < \varepsilon) \to \frac{1}{2}, n \to \infty$.

Hence it is possible to determine if the sequence $a_{i_1}, \ldots, a_{i_k}$ is frequent or rare only passing over the whole original database or using re-sampling technique

described in the following chapter. On the other hand, p is close to the empirical frequency $\bar{p}_n$ when n is sufficiently large, because again according to the central limit theorem for any $\mu > 0$

$$P(|\bar{p}_n - p| > \mu) \to 0, n \to \infty.$$

The probability of the event $\bar{p}_n \in (\varepsilon - \delta, \varepsilon + \delta)$ can be reduced when reducing δ, but then the first and the second type classification error probabilities are increasing. They can be reduced increasing n. So the compatibility between δ and n is mandatory, and their relation can be expressed as $\delta\sqrt{n} = \text{const}$.

Example. Assume that $\delta = 0{,}05$; $\varepsilon = 0{,}2$; $n = 100$. According to (5), the error probability that the rare sequence is assigned to the class of frequent sequences is less than

$$1 - \Phi\left(\delta \frac{\sqrt{n}}{\sqrt{\varepsilon(1-\varepsilon)}}\right) = 1 - \Phi(1{,}25) \approx 0{,}1056.$$

According to (6), the error probability that the frequent sequence is assigned to class of the rare sequences is less than

$$\Phi\left(-\delta \frac{\sqrt{n}}{\sqrt{\varepsilon_1(1-\varepsilon_1)}}\right) = \Phi\left(-2\delta\sqrt{n}\right) = \Phi(-1{,}0) \approx 0{,}1587.$$

Assume that $\delta = 0{,}05$; $\varepsilon = 0{,}2$; $n = 400$. According to (5), the error probability that the rare sequence is assigned to the class of frequent sequences is less than

$$1 - \Phi\left(\delta \frac{\sqrt{n}}{\sqrt{\varepsilon(1-\varepsilon)}}\right) = 1 - \Phi(2{,}5) \approx 0{,}0062.$$

According to (6), the error probability that the frequent sequence is assigned to the class of rare sequences is less than

$$\Phi\left(-\delta \frac{\sqrt{n}}{\sqrt{\varepsilon_1(1-\varepsilon_1)}}\right) = \Phi\left(-2\delta\sqrt{n}\right) = \Phi(-2{,}0) \approx 0{,}0228.$$

3.4. Novel approximate method: Multiple Re-sampling Method (MRM)

RSM method classifies all sequences into the following 3 classes:

1. **Frequent sequences**. The sequence $a_{i_1}, \dots, a_{i_k}$ is assigned to the class of frequent sequences, if $\bar{p}_n(a_{i_1}, \dots, a_{i_k}) \geq \varepsilon + \delta$.
2. **Rare sequences**. The sequence $a_{i_1}, \dots, a_{i_k}$ is assigned to the class of rare sequences, if $\bar{p}_n(a_{i_1}, \dots, a_{i_k}) \leq \varepsilon - \delta$;
3. **Intermediate sequences**. The sequence $a_{i_1}, \dots, a_{i_k}$ is assigned to the class of intermediate sequences, if $\bar{p}_n(a_{i_1}, \dots, a_{i_k}) \in (\varepsilon - \delta, \varepsilon + \delta)$.

Intermediate sequence class is created in RSM method because the classification error for all sequences that belong to this class is very large and can be close to $\frac{1}{2}$. The classification error probability depends on how much the true frequency p differs from ε and according to the central limit theorem when $p = \varepsilon$, then the incorrect classification probability can be close to $\frac{1}{2}$:

$$P(\bar{p}_n \geq \varepsilon) \to \frac{1}{2}, n \to \infty \text{ and } P(\bar{p}_n < \varepsilon) \to \frac{1}{2}, n \to \infty.$$

For determining whether the intermediate sequence is frequent or rare, the full original database review could be performed, however in many cases due to the volume and variety of big data it is hardly possible. Another approach proposed in MRM method is to reduce the number of intermediate sequences to minimum using re-sampling strategy, that creates multiple random samples of the original database, analyses them using RSM method and based on the analysis results, decides whether the sequence is frequent or rare. Assume that we have created $h > 1$ random samples of the original database S, where $n \in [1, N-1]$ is the size of the random sample of the original database and for each sequence we would have such results:

1. h_1 – number of times the sequence was identified as frequent;

2. h_2 – number of times the sequence was identified as rare;
3. h_3 – number of times the sequence was identified as intermediate.

Note that for analysis of all created h random samples, same user-specified parameters should be maintained: $\varepsilon \in (0,1)$ – the minimum support, $\delta > 0$ such that $(0 < \varepsilon - \delta < \varepsilon + \delta < 1)$ – the classification threshold, and $n \in [1, N-1]$ – the size of the random sample $\bar{S_n}$ that will be used in all $h > 1$ random samples.

The decision if the sequence is frequent or rare in re-sampling strategy is made according to max (h_1, h_2, h_3):

1. The sequence is frequent, if $h_1 > h_2$ and $h_1 > h_3$;
2. The sequence if rare, if $h_2 > h_1$ and $h_2 > h_3$.

If none of the above options are satisfied and $max\ (h_1, h_2, h_3) = h_3$, then the number of interim sequences h_3 is eliminated and the sequence is considered to be frequent if $h_1 > h_2$; otherwise rare, if $h_2 > h_1$. An exceptional case could arise if $h_1 = h_2$ and the MRM considers such sequence frequent. This is done in order to avoid losing potential candidates to frequent sequences when generating the candidate sequences of length k are only generated from shorter length $k-1$ frequent sub-sequences (according to GSP algorithm the candidate sequences of length k are only generated from shorter length $k-1$ frequent sub-sequences). Note that the classification error in case $max\ (h_1, h_2, h_3) = h_3$ and $h_1 = h_2$ is already estimated using the statistical methods, therefore this decision is not impacting the classification error.

Let us analyze the MRM method classification into frequent and rare sequences in Table 4 (MRM method analyzes $h = 100$ random samples):

Table 4. MRM method classification example

Sequence	h_1 (frequent)	h_2 (rare)	h_3 (intermediate)	MRM classification
AA	50	14	36	Frequent
AB	30	38	32	Rare
AC	11	29	60	Rare
BA	24	24	52	Frequent

Using the re-sampling strategy, the final classification of frequent and rare sequences could be completed without consulting with the original database and making full original database scan. The experimental results with and without re-sampling strategy are displayed in the following chapters together with recommendations on the number of re-samplings h.

3.5. Novel approximate method: Markov Property Based Method (MPBM)

Markov property bases method (MPBM) makes several passes over the original database and identifies higher level approximate frequent sequences using exactly identified frequent sequences of the lower levels. For the 1st order Markov model, in order to find all approximate frequent sequences in the original databases, it is sufficient to make 2 passes over the original database (for the 2nd order Markov model it is sufficient to make 3 passes, etc.). The detailed method description is given below.

Let the original database be a set $S = \{x_1, x_2, \dots, x_N\}$ whose elements x_i can take M different values in a set $V = \{\alpha_1, \alpha_2, \dots, \alpha_M\}$. A sequence $(a_{i_1}, \dots, a_{i_m})$ *is* called frequent if (# is a number):

$$\frac{1}{N-m+1}\#\{j \in \{1, \dots, N-m+1\}: x_j = \alpha_{i_1}, x_{j+1} = \alpha_{i_2}, \dots, x_{j+m-1} = \alpha_{i_m}\} \geq \varepsilon,$$

where $\varepsilon \in (0,1)$ is a given threshold also known as a minimum support.

We will interpret the data base as a realization of a stationary ergodic V-valued process.

Assume $X_n, -\infty < n < \infty$, is a stationary finite set V-valued stochastic process: for each $n, m, t_1, \dots, t_n$ and $\alpha_{i_1}, \dots, \alpha_{i_n} \in V$,

$$P\left(X_{t_1} = \alpha_{i_1}, \dots, X_{t_n} = \alpha_{i_n}\right) = P(X_{t_1+m} = \alpha_{i_1}, \dots, X_{t_n+m} = \alpha_{i_n})$$

We define the empirical frequencies:

$$\bar{p}\left(\alpha_{i_1}, \dots, \alpha_{i_m}\right) = \frac{1}{N-m+1} \#\{j \in \{1, \dots, N-m+1\} : X_j = \alpha_{i_1}, X_{j+1} = \alpha_{i_2}, \dots, X_{j+m-1} = \alpha_{i_m}\}.$$

According to the Theorem 6.1 in (Varadhan, 2001), $\bar{p}\left(\alpha_{i_1}, \dots, \alpha_{i_m}\right)$ has a limit a.s. and in $L_1(P)$. Because of the ergodic property, the limit is $p\left(\alpha_{i_1}, \dots, \alpha_{i_m}\right) = P\left\{X_0 = \alpha_{i_1}, X_1 = \alpha_{i_2}, \dots, X_{m-1} = \alpha_{i_m}\right\}$:

$$\bar{p}\left(\alpha_{i_1}, \dots, \alpha_{i_m}\right) \to p\left(\alpha_{i_1}, \dots, \alpha_{i_m}\right) \text{ as } N \to \infty. \qquad (7)$$

In the following two Remarks, 1 and 2, we discuss some sufficient conditions of ergodicity.

Definition 1. The whole space Ω and the empty set Φ are in D (class of sets). For any two sets A and B in D, the sets $A \cup B$ and $A \cap B$ are again in D. If $A \in B$, then the complement AC is again in D. The class of sets D that satisfies these properties is called a field.

Definition 2. The class D, in addition to being a field is assumed to be closed under countable union (or equivalently, countable intersection); i.e. if $A_n \in D$ for every n, then $A = \cup_n A_n \in D$. Such a class D is called a σ-field. The 'probability' itself is presumed to be defined on a σ-field D.

Remark 1. X_n is ergodic if infinitely remote past $F_{-\infty} = \bigcap_n \sigma(X_k, k \leq n)$ and infinitely remote future are independent $F_{\infty} = \bigcap_n \sigma(X_k, k \geq n)$: for any $A \in F_{-\infty}$, $B \in F_{\infty}$,

$$P(A \cap B) = P(A)P(B).$$

Here $\sigma(X_k, k \leq n)$ is σ-field generated by $X_k, k \leq n$. Remark 1 follows from the proof of Theorem 6.1 in (Varadhan, 2001).

Now, let us assume in addition that X_n is Markov (it is meant 1st order Markov). It is characterized by transition probabilities:

$$p(\alpha; \beta) = P(X_{n+1} = \beta | X_n = \alpha), \alpha, \beta \in V$$

and $\mu(\alpha) = P(X_n = \alpha)$. By definition of the conditional probability, $p(\alpha, \beta) = P(X_n = \alpha, X_{n+1} = \beta) = p(\beta; \alpha)\mu(\alpha)$. Note the difference: $p(\alpha, \beta)$is joint probability and $p(\alpha; \beta)$ is transition (conditional) probability. Also, obviously,

$$\mu(\beta) = \sum_{\alpha} p(\beta; \alpha)\mu(\alpha), \beta \in V \qquad (8)$$

A probability measure τ on V is called invariant if (8) holds with μ replaced by τ.

Remark 2. According to Chapter 6.3 in (Varadhan, 2001), X_n is ergodic if it has a unique invariant measure μ. For example, by Theorem 1.10.2 in (Norris, 1997), invariant measure is unique if X_n is irreducible and positive recurrent:

Irreducible: for all $\alpha, \beta \in V$, there exists $m = m(\alpha, \beta)$such that X_n can reach β from α in m steps with positive probability.

Positive recurrent: For any $\alpha \in V$ the expected return time to α is finite.

Let's assume, that $X_n, n = 1,2, ...$ is a stationary 1st order Markov process. We denote:

$$p(\alpha_{i_1}\alpha_{i_2}\dots\alpha_{i_k}) = P\{X_n = \alpha_{i_1}, X_{n+1} = \alpha_{i_2}, \dots, X_{n+k-1} = \alpha_{i_k}\}, k = 1,2,\dots$$

Let's assume we know $p(\alpha_i), p(\alpha_i\alpha_j)$, $i,j = 1,\dots,M$.

LEMMA 1. Let $X_n, n = 1,2,\dots$ is a stationary ergodic 1st order Markov process. Then approximating the original database by a stationary 1st order Markov process (m=1) for each $\alpha_{i_1},\dots,\alpha_{i_m} \epsilon\ V$, we derive

$$p(\alpha_{i_1},\dots,\alpha_{i_m}) = \frac{p(\alpha_{i_1,}\alpha_{i_2})p(\alpha_{i_2},\alpha_{i_3})\dots p(\alpha_{i_{m-1}},\alpha_{i_m})}{p(\alpha_{i_2})\dots p(\alpha_{i_{m-1}})}, m \geq 3.$$

Proof. We will be using Markov property for proving this lemma. According to Markov property for each $n \epsilon \{1,\dots,N\}$, k is the order of Markov process and $\alpha_{i_1},\dots,\alpha_{i_n} \epsilon\ V$

$$P\{X_n = \alpha_{i_n} | X_{n-1} = \alpha_{i_{n-1}},\dots,X_1 = \alpha_{i_1}\} = P\{X_n = \alpha_{i_n} | X_{n-1} = \alpha_{i_{n-1}},\dots,X_{n-k} = \alpha_{i_{n-k}}\}$$

By the definition of conditional probability:

$$p(\alpha_{i_1},\alpha_{i_2},\alpha_{i_3}) = P\{X_1 = \alpha_{i_1}, X_2 = \alpha_{i_2}, X_3 = \alpha_{i_3}\} =$$

$$= P\{X_3 = \alpha_{i_3} | X_2 = \alpha_{i_2}, X_1 = \alpha_{i_1}\} P\{X_2 = \alpha_{i_2}, X_1 = \alpha_{i_1}\} =$$

$$= P\{X_3 = \alpha_{i_3} | X_2 = \alpha_{i_2}\} P\{X_2 = \alpha_{i_2}, X_1 = \alpha_{i_1}\} =$$

$$= \frac{P\{X_3 = \alpha_{i_3}, X_2 = \alpha_{i_2}\}}{P\{X_2 = \alpha_{i_2}\}} P\{X_2 = \alpha_{i_2}, X_1 = \alpha_{i_1}\} = \frac{p(\alpha_{i_1,}\alpha_{i_2})p(\alpha_{i_2},\alpha_{i_3})}{p(\alpha_{i_2})}$$

For simplicity the case of $m = 3$ is analyzed, and when $m > 3$ the proof is analogous. So, approximating the database by a stationary 1st order process, we have to know only $p(\alpha_{i_1})$ and $p(\alpha_{i_1},\alpha_{i_2})$ for all $\alpha_{i_1},\alpha_{i_2} \epsilon\ V$. The lemma is proved.

On the other hand, due to (7),

$$\bar{p}(\alpha_{i_1}, \dots, \alpha_{i_m}) \approx \frac{\bar{p}(\alpha_{i_1,}\alpha_{i_2})\bar{p}(\alpha_{i_2,}\alpha_{i_3})\dots\bar{p}(\alpha_{i_{m-1}},\alpha_{i_m})}{\bar{p}(\alpha_{i_2})\bar{p}(\alpha_{i_3})\dots\bar{p}(\alpha_{i_{m-1}})} \qquad (9)$$

for $m \geq 3$ and sufficiently large N. Therefore, computation of $\bar{p}(\alpha_{i_1})$ and $\bar{p}(\alpha_{i_1,}\alpha_{i_2})$ for all $\alpha_{i_1}, \alpha_{i_2} \epsilon\ V$ enables us to compute approximately all the frequencies $\bar{p}(\alpha_{i_1}, \dots, \alpha_{i_m})$, $m \geq 3$. For mining frequent sequences the formula (9) should be used together with the following property of frequent sequences: if $\bar{p}(\alpha_{i_1}, \dots, \alpha_{i_m}) \geq \varepsilon$ and $(\alpha_{j_1}, \dots, \alpha_{j_l})$ is a subset of $(\alpha_{i_1}, \dots, \alpha_{i_m})$, then $\bar{p}(\alpha_{j_1}, \dots, \alpha_{j_l}) \geq \bar{p}(\alpha_{i_1}, \dots, \alpha_{i_m}) \geq \varepsilon$; in particular, all the multipliers in the right hand side of (9) should be not less than ε.

LEMMA 2. Let $X_n, n = 1,2, \dots$ is a stationary ergodic 2nd order Markov process. Then

$$p(\alpha_{i_1}, \dots, \alpha_{i_m}) = \frac{p(\alpha_{i_1,}\alpha_{i_2}, \alpha_{i_3})p(\alpha_{i_2}, \alpha_{i_3}, \alpha_{i_4}) \dots p(\alpha_{i_{m-2}}, \alpha_{i_{m-1}}, \alpha_{i_m})}{p(\alpha_{i_2,}\alpha_{i_3})p(\alpha_{i_3}, \alpha_{i_4}) \dots p(\alpha_{i_{m-2}}, \alpha_{i_{m-1}})}, m \geq 4$$

Proof. As per Lemma 1 proof, we will use Markov property and definition of the conditional probability:

$$\begin{aligned} p(\alpha_{i_1}, \alpha_{i_2}, \alpha_{i_3}, \alpha_{i_4}) = \\ &= P\{X_{n+3} = \alpha_{i_4} | X_n = \alpha_{i_1}, X_{n+1} = \alpha_{i_2}, X_{n+2} = \alpha_{i_3}\} P\{X_n \\ &= \alpha_{i_1}, X_{n+1} = \alpha_{i_2}, X_{n+2} = \alpha_{i_3}\} \\ &= P\{X_{n+1} = \alpha_{i_2} | X_{n+2} = \alpha_{i_3}, X_{n+3} = \alpha_{i_4}\} \frac{p(\alpha_{i_1,}\alpha_{i_2}, \alpha_{i_3})}{p(\alpha_{i_2}, \alpha_{i_3})} \\ &= \frac{p(\alpha_{i_1,}\alpha_{i_2}, \alpha_{i_3})p(\alpha_{i_2}, \alpha_{i_3}, \alpha_{i_4})}{p(\alpha_{i_2}, \alpha_{i_3})} \end{aligned}$$

Therefore, computation of $\bar{p}(\alpha_{i_1})$, $\bar{p}(\alpha_{i_1,}\alpha_{i_2})$ and $\bar{p}(\alpha_{i_1,}\alpha_{i_2}, \alpha_{i_3})$ for all $\alpha_{i_1}, \alpha_{i_2}, \alpha_{i_3} \epsilon\ V$ enables us to compute approximately all the frequencies $\bar{p}(\alpha_{i_1}, \dots, \alpha_{i_m})$, $m \geq 4$.The lemma is proved.

Again according to theorem (7),

$$\bar{p}(\alpha_{i_1}, \dots, \alpha_{i_m}) \approx \frac{\bar{p}(\alpha_{i_1,}\alpha_{i_2}, \alpha_{i_3})\bar{p}(\alpha_{i_2}, \alpha_{i_3}, \alpha_{i_4}) \dots \bar{p}(\alpha_{i_{m-2}}, \alpha_{i_{m-1}}, \alpha_{i_m})}{\bar{p}(\alpha_{i_2,}\alpha_{i_3})\bar{p}(\alpha_{i_3}, \alpha_{i_4}) \dots \bar{p}(\alpha_{i_{m-2}}, \alpha_{i_{m-1}})}$$

for $m \geq 4$ and sufficiently large N. Similar procedure can be derived for the approximation of the database by stationary *k*-th order Markov process.

The rate of convergence could be estimated using the following approach. Let

$$\rho(k, \alpha_{i_1}, \dots, \alpha_{i_m}) = P(X_1 = \alpha_{i_1}, X_{1+k} = \alpha_{i_1}, \dots, X_m = \alpha_{i_m}, X_{m+k} = \alpha_{i_m})$$

Then a direct computation shows that

$$E\left(\left[\tilde{p}(\alpha_{i_1}, \dots, \alpha_{i_m}) - p(\alpha_{i_1}, \dots, \alpha_{i_m})\right]^2\right) \leq \sum_{k \geq 1} 2(N + m - 1)^2 \rho(k, \alpha_{i_1}, \dots, \alpha_{i_m})$$

According to (Billingsley, 1961): if $X_n$is stationary Markov ergodic, then

$$(N-1)\frac{[\tilde{p}(\beta, \alpha) - p(\beta, \alpha)]^2}{p(\beta, \alpha)}$$

have asymptotically χ-square distribution. By Lemma 3.2 in (Billingsley, 1961),

$$(N-1)E([\tilde{p}(\alpha) - p(\alpha)]^2) \leq C$$

where the constant C is computed in (Billingsley, 1961). This provides the rates of convergence for the MPBM method as well.

3.6. Conclusions

Approximate probabilistic frequent sequence mining methods could deliver the acceptable results with right balance between the computing speed and precision when dealing with significantly growing amount of digital

information in big data; whereas the exact frequent sequence mining methods work under the time constraints and cannot guarantee the required speed by various real-world applications.

Novel approximate Random Sampling Method (RSM), proposed in this thesis, brings a significant benefit among existing approximate frequent sequence mining methods by providing a theoretical estimation of error probabilities. In RSM method it is proved that the error probabilities made by this method could be theoretically estimated using standard statistical methods and central limit theorem. The theoretical estimation allows using the proposed method without any extensive empirical experiments used in other state-of-the-art approximate methods to evaluate the performance of the method. It also gives information to the data analyst about what error probability could be expected beforehand and does not require any empirical experiments to define the method precision. Another benefit that RSM method provides is that it does not require reading the original database at all and the recommendations are given to select parameters using by the algorithm so that even more precise results are achieved.

Markov Property Based Method (MPBM) proposed in this thesis does not satisfy the time constraints as it requires reading the original database at least several times (the number of times equals to the order of the Markov process) and it delivers similar precision results as Multiple Re-sampling Method (MRM). So the approximate frequent pattern mining algorithms that require reading the entire original database do not satisfy the time constraints required by many real-world applications.

Chapter 4: Experimental evaluation of the proposed methods and comparison with other existing algorithms

The performance studies for proposed approximate methods RSM, MRM and MPBM were performed on real databases by varying different database parameters and by comparing it with other frequent sequence mining algorithms like GSP, ApproxMAP and ProMFS. All experiments were performed on a laptop with 2.4 GHz Intel Celeron processor, and with 4GB RAM memory, running Windows. All methods were implemented using Java programming language.

4.1. Datasets used for performance evaluation

The performance studies were performed on the real financial database and synthetic genetic database:

1. The financial database consists of the foreign exchange currency pair EUR-USD hourly data from 03/01/2000 till 01/05/2013 (data is taken from *Online Trading Platform MetaTrader 4 History Center* (MetaTrader 4 software, retrieved 2013)). The financial database consists of $N = 5397843$ elements with possible values {A, B, C}, which indicate if the currency exchange rate is growing, falling or being the same as previous hour:
   (a) A – if the i-th hourly closing exchange rate C_i is higher than opening exchange rate O_i (i.e. $C_i > O_i$), then $S_i = A$;
   (b) B – if the i-th hourly closing exchange rate C_i is less than opening exchange rate O_i (i.e. $C_i < O_i$), then $S_i = B$;
   (c) C – if the i-th hourly closing exchange rate C_i is equal to the opening exchange rate O_i (i.e. $C_i = O_i$), then $S_i = C$.

2. The genetic database consists of $N = 66326866$ elements with possible values {A, T, C, G}. These values indicate nucleobases are nitrogen-containing biological compounds found within DNA. The primary nucleobases in DNA are cytosine, guanine, adenine, thymine, abbreviated as C, G, A, T respectively. The sample data was collected using DNA Baser Sequence Assembler software (DNA Baser, retrieved in 2013).

4.2. Parameter selection for proposed approximate methods

For the original database S analysis we used the exact frequent sequence mining algorithm GSP and proposed approximate RSM, MRM and MPBM methods to identify frequent sequences in the original databases. The following user-specified parameters were used for testing proposed approximate methods:

- Minimum support threshold $0 < \varepsilon < 1$ that is used to determine if the sequence is frequent or rare. The sequence is considered to be frequent if $\bar{p}_n \geq \varepsilon$; otherwise it is rare. Minimum support threshold is used in all methods: GSP, RSM, MRM, MPBM, ProMFS;
- The size of the random sample n created from the original database for RSM and MPBM algorithms;
- The classification error threshold $\delta > 0$ such that $(0 < \varepsilon - \delta < \varepsilon + \delta < 1)$ used for approximate RSM and MRM methods;
- The number of re-samplings $h = 2, \dots, 200$ used by approximate MRM method;
- The order of Markov process $m = 1, 2, 3$ used in the approximate MPBM method.

4.3. Performance evaluation for approximate RSM method

The exact GSP algorithm and approximate RSM method will be used for analysis of the original database S and the true frequencies (determined by GSP algorithm) are compared with empirical frequencies (determined by approximate RSM method). Assume that sequence is frequent, if it's frequency is not less than 0,08 (the minimum support threshold is $\varepsilon = 0{,}08$).

First of all, the original database is analyzed using exact GSP algorithm and frequent sequences determined precisely. Then using the RSM method the frequent sequences in the random sample are determined (the random sample sizes $n = 500$, $n = 1000$ and $n = 2000$; the classification error threshold $\delta = 0{,}02$).

According to (5), the first type classification error probability (the rare sequence is assigned to the class of frequent sequences) is given by:

$n = 500$: $1 - \Phi\left(\delta\frac{\sqrt{n}}{\sqrt{\varepsilon(1-\varepsilon)}}\right) = 1 - \Phi(1.6485) \approx 0.0496;$

$n = 1000$: $1 - \Phi\left(\delta\frac{\sqrt{n}}{\sqrt{\varepsilon(1-\varepsilon)}}\right) = 1 - \Phi(2.3313) \approx 0.0099;$

$n = 2000$: $1 - \Phi\left(\delta\frac{\sqrt{n}}{\sqrt{\varepsilon(1-\varepsilon)}}\right) = 1 - \Phi(3.2969) \approx 0.0005.$

According to (6), the second type classification error probability (the rare sequence is assigned to the class of frequent sequences) is given by:

$n = 500$: $\Phi\left(-\delta\frac{\sqrt{n}}{\sqrt{\varepsilon_1(1-\varepsilon_1)}}\right) = \Phi\left(-2\delta\sqrt{n}\right) = \Phi(-0.8944) \approx 0.1855;$

$n = 1000$: $\Phi\left(-\delta\frac{\sqrt{n}}{\sqrt{\varepsilon_1(1-\varepsilon_1)}}\right) = \Phi\left(-2\delta\sqrt{n}\right) = \Phi(-1.2649) \approx 0.1030;$

$n = 2000$: $\Phi\left(-\delta\frac{\sqrt{n}}{\sqrt{\varepsilon_1(1-\varepsilon_1)}}\right) = \Phi\left(-2\delta\sqrt{n}\right) = \Phi(-1.7889) \approx 0.0368$.

The experiment results of the precision of the RSM method are shown in the Table 5.

Table 5. RSM method results for the random sample sizes $n = 500$, $n = 1000$ and $n = 2000$ and the classification error threshold $\delta = 0{,}02$

Data	Metrics	GSP algorithm	Random Sampling Method (RSM)		
			n=500	*n*=1000	*n*=2000
Foreign Exchange database	Number of frequent sequences identified	265	192	193	212
	Number of rare sequences identified as frequent	0	27	25	17
	Number of frequent sequences identified as rare	0	16	9	5
	Number of intermediate sequences identified	0	231	163	86
DNA database	Number of frequent sequences identified	144	114	126	149
	Number of rare sequences identified as frequent	0	29	12	4
	Number of frequent sequences identified as rare	0	11	9	6
	Number of intermediate sequences identified	0	238	126	35

It is important to note that when the random sample size n is increasing, then the number of intermediate sequences and incorrectly classified sequences is decreasing. The number of intermediate sequences can be reduced when reducing δ, but then the first and the second type classification error probabilities are increasing. They can be reduced increasing n. So the compatibility between δ and n is mandatory, and their relation can be defined as $\delta\sqrt{n} = \text{const}$. Figure 13 demonstrates the dependency between the random sample size n and the classification error for fixed $\varepsilon = 0.08$ and the classification error threshold $\delta = 0.001$ and 0.005 .

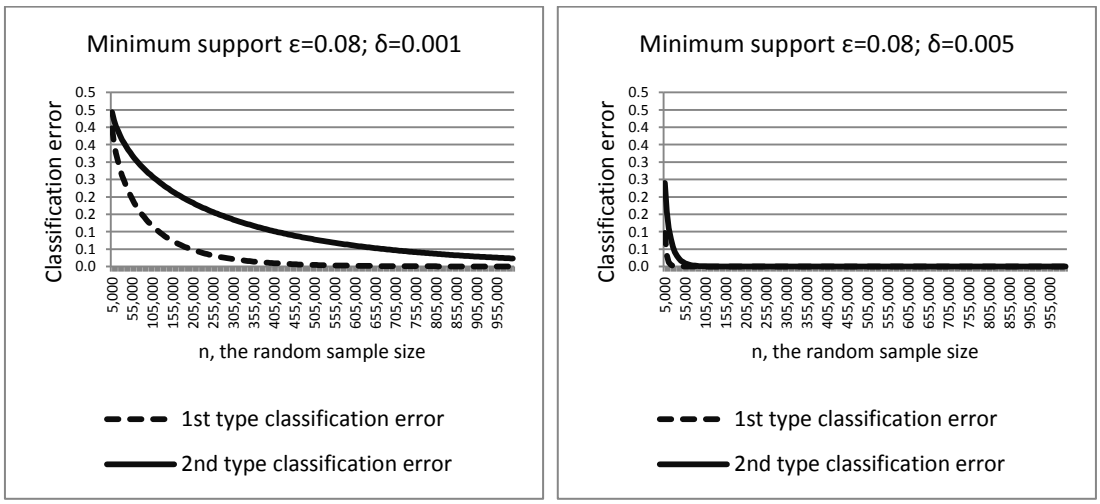


Figure 13: The 1st and 2nd type classification errors for $\varepsilon = 0.08; \delta = 0.001$ and 0.005.

The Figures 14, 15 and 16 demonstrate how the first and second type classification error probabilities decrease when the random sample size $n = 1000$, 10000 and 100000 is increasing, even if the classification error threshold is very low $\delta < 0.01$. Therefore when the random sample size n is high, then it is recommended to keep the classification error threshold as low as possible in order to have a low number of intermediate sequences. The classification of the intermediate sequences into frequent and rare might lead to incorrect classification, because the error probability could be close to $\frac{1}{2}$.

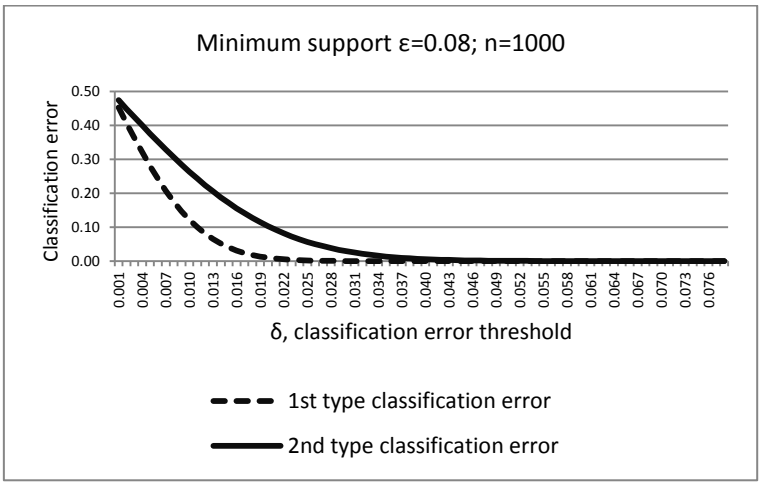


Figure 14: The 1st and 2nd type classification errors for $\varepsilon = 0.08; n = 1000$.

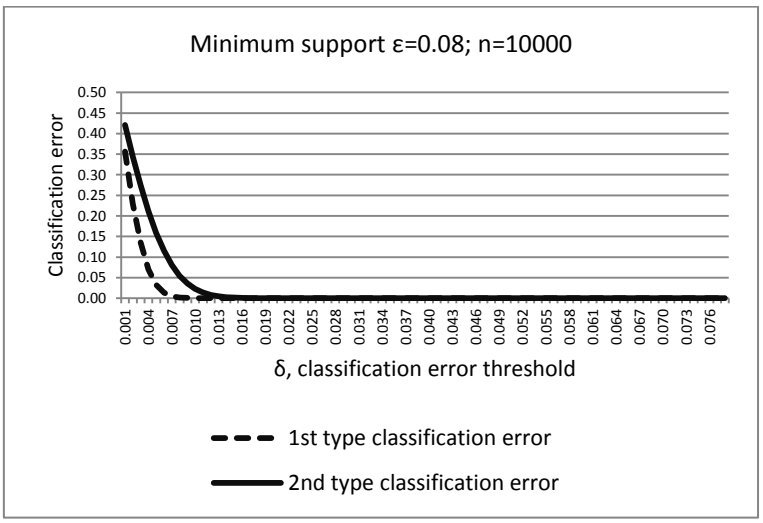


Figure 15: The 1st and 2nd type classification errors for $\varepsilon = 0.08; n = 10000$.

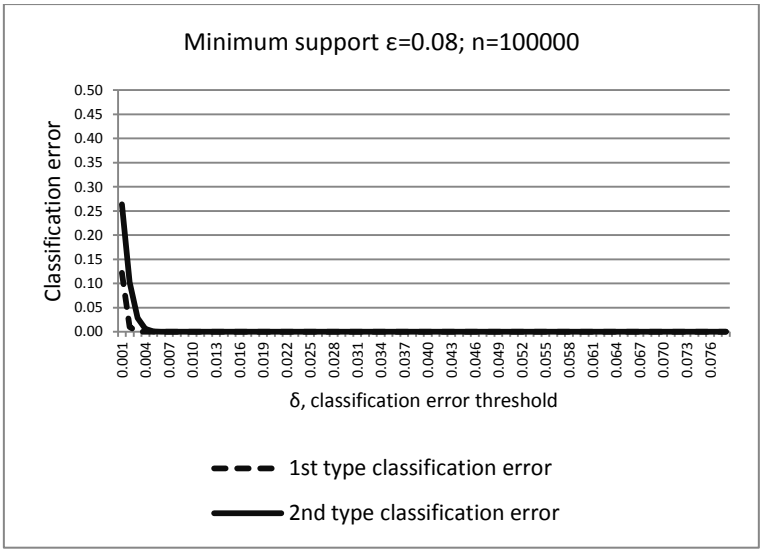


Figure 16: The 1st and 2nd type classification errors for $\varepsilon = 0.08; n = 100000$.

4.4. Performance evaluation for approximate MRM method

For evaluating the results of the approximate MRM method we use the following parameters as for RSM method: the random sample sizes $n = 1000$ and $n = 2000$; the minimum support threshold is $\varepsilon = 0{,}08$ and the classification error threshold $\delta = 0{,}02$ and will compare the precision results with it.

In order to decide how many re-samplings would be required to deliver an acceptable result, the additional test is performed on the fixed size of the random sample $n = 5000$ in the Foreign Exchange database and the number of re-samples was $h = 2, \dots, 300$.

For each sequence that has been identified as intermediate during the re-sampling we calculate $max\ (h_1, h_2, h_3)$ and eliminate all frequent sequences that satisfy the condition (1) $max\ (h_1, h_2, h_3) = h_1$ and also eliminate all rare sequences that satisfy the condition (2) $max\ (h_1, h_2, h_3) = h_2$. All the rest sequences are intermediate sequences that satisfy the condition (3) $max\ (h_1, h_2, h_3) = h_3$ and taking the decision if the sequence is frequent or rare has higher error probability compared to decisions made in (1) and (2). The number of intermediate sequences h_3 that satisfy the condition (3) is decreasing when the number of re-samples h increases; however it reaches the minimum at $h = 30$ and stabilizes as shown in the Figure 17. For further experiments we will use the number of re-samplings equal to $h = 30$.

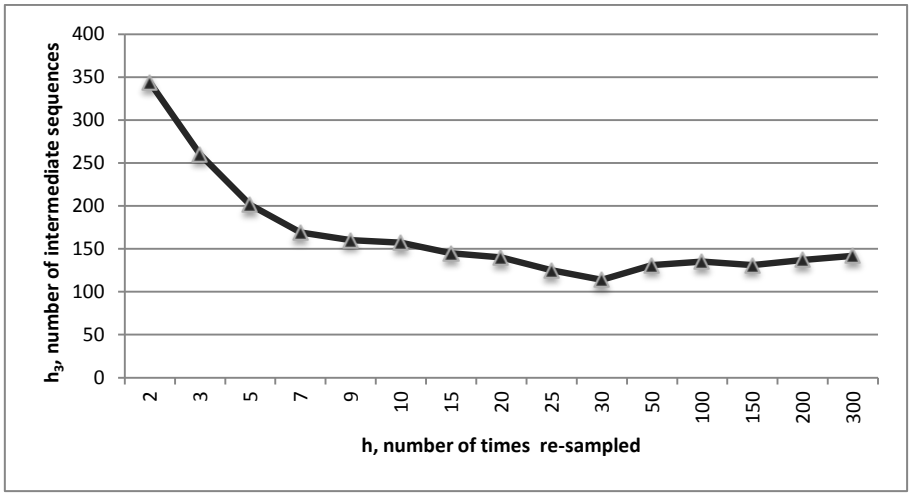


Figure 17. The number of intermediate sequences h_3 after a number of re-samplings $h = 2, \dots, 300$.

Table 6. The results of MRM and RSM methods for random sample sizes $n = 1000$ and $n = 2000$; the classification error threshold $\delta = 0{,}02$

Data	Metrics	GSP algorithm	Random Sampling Method (RSM)		Multiple Re-sampling Method (MRM)	
			n=1000	*n*=2000	*n*=1000	*n*=2000
Foreign Exchange database	Number of frequent sequences identified	265	193	212	261	263
	Number of rare sequences identified as frequent	0	25	17	30	22
	Number of frequent sequences identified as rare	0	9	5	13	7
	Number of intermediate sequences identified	0	163	86	0	0
DNA database	Number of frequent sequences identified	144	126	149	139	146
	Number of rare sequences identified as frequent	0	12	4	15	6
	Number of frequent sequences identified as rare	0	9	6	13	8
	Number of intermediate sequences identified	0	126	35	0	0

4.5. Performance evaluation for approximate MPBM method

For the precise identification of frequent patterns in the original database S the exact GSP algorithm will be used. The same minimum support threshold $\varepsilon = 0{,}08$ will be used as in RSM method in the Chapter 3.6.3.

We will use 3 different orders of Markov process m = 1, 2 and 3 in the approximate MPBM method and compare the results delivered by MPBM method with another approximate RSM method. The same parameters will be used in the RSM method (the random sample sizes $n = 1000$ and $n = 2000$; the classification error threshold $\delta = 0{,}02$) in order to compare it with MPBM method.

Table 7. The results of MPBM (m = 1, 2 and 3), RSM ($n = 1000$ and $n = 2000$; $\delta = 0{,}02$) and MRM ($h = 30$) methods

Data	Metrics	GSP algorithm	Markov Property Based Method (MPBM)			Random Sampling Method (RSM)		Multiple Re-sampling Method (MRM)	
			m=1	*m=2*	*m=3*	*n*=1000	*n*=2000	*n*=1000	*n*=2000
Foreign Exchange database	Number of frequent sequences identified	265	268	266	266	193	212	261	263
	Number of rare sequences identified as frequent	0	6	4	2	25	17	30	22
	Number of frequent sequences identified as rare	0	3	3	1	9	5	13	7
	Number of intermediate sequences identified	0	0	0	0	163	86	0	0
DNA database	Number of frequent sequences identified	144	144	146	146	126	149	139	146
	Number of rare sequences identified as frequent	0	0	20	8	12	4	15	6
	Number of frequent sequences identified as rare	0	0	18	6	9	6	13	8
	Number of intermediate sequences identified	0	0	0	0	126	35	0	0

The experiment results show that the MRM method showed the best results in terms of precision on both Foreign Exchange database and DNA database; in the second place there is the 3rd order MPBM method; however it requires reading the original database 4 times, which is very time consuming. RSM method does not read the original database, only the random sample of the original database; however it has a quite large number of intermediate sequences and according to the precision is outperformed by MPBM and MRM methods. In order to make the classification if such sequence is rare or frequent, it is recommended to use MRM method to minimize the number of intermediate sequences to minimum. MRM method has overall best results in terms of the precision, but it is requiring h times more time to deliver the classification results compared to RSM, because it analyzes $h = 30$ ramdom samples until it delivers the classification result.

The evaluation of computation speed on DNA database of the proposed methods demonstrated that RSM method has overall best results in terms of

computation speed (see Figure 18). MRM method ($h = 30$) required h times more computation time compared to the RSM method. MPBM and ProMFS methods required a much longer computation time, because they require reading the entire original database, which is a time consuming task.

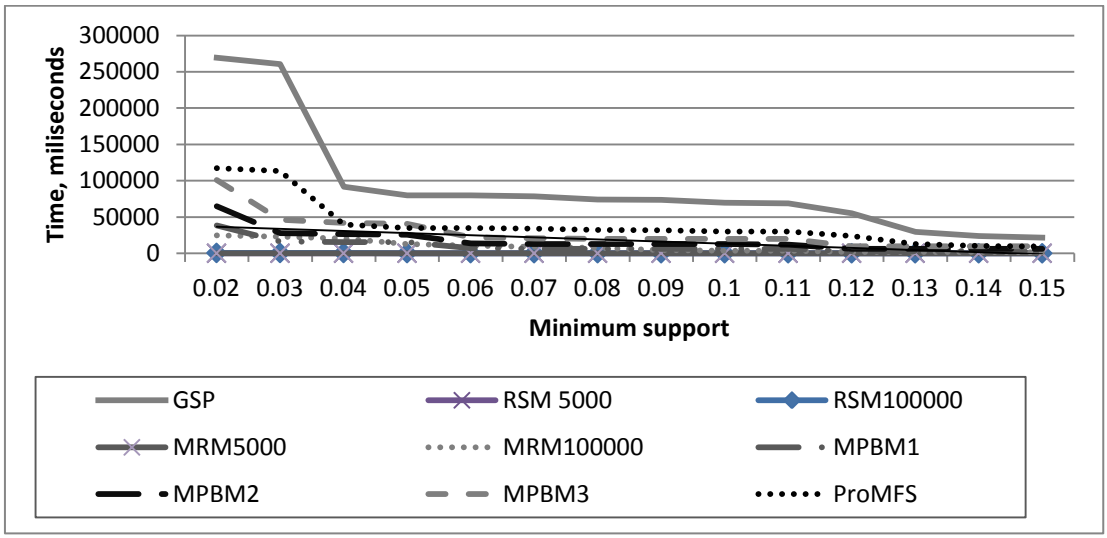


Figure 18. Proposed approximate methods computation speed evaluation ($\delta = 0.1, h = 30$).

4.6. Conclusions

Approximate probabilistic frequent sequence mining methods could deliver the acceptable results in real-world applications where the computing speed is more important than the precision.

Novel approximate Random Sampling Method (RSM), proposed in this thesis, brings a significant benefit among existing approximate frequent sequence mining methods by providing a theoretical estimation of error probabilities. The theoretical estimation allows using the proposed method without any empirical experiments or observations on different databases used in other state-of-the-art approximate methods to evaluate the performance of the method. RSM method does not require reading the original database at all and the recommendations are given to select parameters using by the algorithm so

that even more precise results are achieved. The evaluation of computation speed demonstrated that RSM method has overall best results in terms of computation speed, whereas Multiple Re-sampling Method (MRM) proposed in this thesis has shown the best overall result in terms of the precision among other approximate algorithms, but it was outperformed by RSM method in terms of the computing speed. MRM method required h times more for computing the results.

Markov Property Based Method (MPBM) proposed in this thesis does not satisfy the time constraints as it requires reading the original database at least several times (the number of times equals to the order of the Markov process) and it delivers similar precision results as Multiple Re-sampling Method (MRM). So the approximate frequent pattern mining algorithms that require reading the entire original database do not satisfy the time constraints required by many real-world applications.

Chapter 5: Visualizing big data: Internet User's behavioural data

5.1. Introduction

The analysis of the online customer shopping behaviour is an important task that allows maximizing the efficiency of advertising campaigns and increasing the return of investment for advertisers. The analysis results of online customer shopping behaviour are usually reviewed and understood by a non-technical person; therefore the results must be displayed in the easiest possible way.

The number of Internet users has increased exponentially around the world. Web applications have been growing explosively as well; people enjoy a wide variety of online services, from e-mail and browsing to information services and search, or collaborative services such as wikis, blogs, and social networks. Online purchasing of goods, both expensive and cheap, has been more and more preferable to a much larger extent in recent years due to convenience, speedy transactions and enhanced shopping experience. Customers use the Internet to buy the product online, to compare prices and products. Europeans spend an average of 24 hours on the Internet each month (IAB digital review: Consumers driving the digital uptake, 2010). Almost all of the services that they use regularly -- from the e-mail, instant messaging, maps to social networks, games, music, video sites to search and price comparison -- are free, funded largely by online advertising.

The analysis of online customers behavioural shopping data is very subjective, because such data usually is both numerical and categorical and the priorities of each feature describing the data are not clearly defined. Our idea is to analyse the data using a number of different visualization methods, to visualize customer behavioural data and to present the visualization results for the decisions on the strategy of the advertising campaign to a non-technical person, usually the media planner. Data on the online customer shopping behaviour is multidimensional and consists of two types: numerical and categorical. Several

multidimensional data visualization methods are applied in displaying online shopping data and their relevance is analyzed. The results of the visual analysis of numerical data are combined with the categorical data. Correctly defined customer clusters can be used to identify up-selling and cross-selling opportunities with existing customers. One can also cluster products that tend to sell together. Clustering and visualization of transactional data is a vital component in the retail industry. The proper strategy of the advertising campaign helps advertisers to stay ahead in highly competitive markets.

In this chapter an approach has been proposed for the visual analysis of the online shopping data. It integrates several multidimensional data visualization methods of different nature. The results of the visual analysis of numerical data are combined with the categorical data values. Based on the visualization results, the decisions on the advertising campaign could be taken in order to increase the return of investment and attract more customers to buy in the online e-shop.

5.2. Customer online shopping behaviour data

All e-shop website customers could be classified into either coming directly to this website or through an advertising campaign. The online advertisements in various Internet channels significantly increase the number of customers of the e-commerce websites, especially if the advertisements announce special sales or discounts. According to "The 2010 Europe Digital Year in Review" (comScore: The 2010 Europe Digital Year in Review, 2011) published by comScore Inc. (a company measuring the digital world performance and newest trends) even 97% of all Internet users were reached by display advertisements in Germany, UK and France. The increased level of engagement of Internet users was shown by rich media advertisements and especially by online video advertisements, where a regular EU Internet user spent approximately 15 hours in 2010 watching online videos on the Internet.

The Interactive Advertising Bureau (IAB) Europe reports that Internet advertising is approaching a 20% share of total advertising spending (IAB digital review: Consumers driving the digital uptake, 2010), almost doubling its share over the last two years. While the previous years saw a contraction in marketing budgets, the digital marketplace in Europe continues growing. With its growth there comes innovation -- from advanced planning tools to sophisticated targeting techniques. Social media grew immensely in 2010 and saturated at least three quarters of individual European markets, whereas the display advertising market continued to grow in 2010, with the social networking category accounting for a rapidly increasing share of impressions. The online advertising market continues evolving, with more advanced advertisements, improved targeting capabilities, and higher quality creativity.

Each advertising campaign captures a huge number of various online data, e.g. referrals, impressions and clicks on advertisements in media websites, page views, products viewed, products purchased, the price paid, etc. All this data is usually measured using browser cookies which are text files stored on the user's computer by the web browser. The online statistics show that 87% of users have accepted the 3rd party cookies (Visitor Browser & Cookie Demographics, retrieved 2013) and this allows analyzing the performance of advertising campaigns and shopping behaviour of different customer segments quite accurately. The customers can come to the e-shop via different channels (campaign traffic and non-campaign related traffic). In campaign traffic, the customers interacted with the advertising campaign material before they came to the advertiser website, e.g. they saw or clicked on the campaign banners or the adwords in the search engines, etc.; whereas in the non-campaign related traffic, the customers did not interact with the campaign material and went straight to the advertiser website. Note that for the non-campaign traffic, it is impossible to capture the same features as for the campaign traffic, e.g. campaign name, advertisement type, advertisement size, etc.

There is a number of different analysis methods performed recently with online data. The online data may be obtained from a clickstream. The clickstream can be defined as a path a customer takes through one or more websites and it can include within-site information such as the pages visited, time spent, etc. Researchers have investigated customer behaviours across websites (Bucklin et al., 2002; Bucklin, Sismeiro, 2003; Park, Fader, 2004) and within a particular website (Goldfarb, 2002; Moe, 2003). Another study aimed at investigating the complexity of online shopping behaviour (Senecal, 2005) from many different online decision-making processes.

There were two advertising campaigns running for a particular advertiser in August 2010 (5361 purchases in total), with various creative material (flash and rich media) shown in the most visited media websites. Also search adwords were shown in search engines (Google, Yahoo, MSN) when customers were searching for specific queries related with the advertiser. The advertiser website was entirely tracked including the purchase amount. Each purchase in the advertiser online shop was defined by 9 different features $x_1,\ x_2 \ldots, x_9$. Five numerical features are detailed below:

- x_1 - campaign channel, which leads customers to the advertiser website: 1 - campaign traffic (customers who came through campaign material), 0 - non-campaign traffic (customers who came directly to the advertiser website);
- x_2 - the advertisement size in cm^2;
- x_3 - minutes from the last interaction: for campaign traffic - it shows the minutes from last interaction with campaign material to the purchase made in the advertiser website (the time from the last moment when the user interacted with the campaign material and the moment of purchasing; not necessary the purchase is performed in the same session as the interaction with campaign material); for non-campaign traffic - it shows the minutes since the customer opened any page in the advertiser website to the purchase without interacting with any campaign material;

- x_4 - the number of interactions before the purchase was made;
- x_5 - the sale amount of the purchase.

Four categorical features are detailed below:

- x_6 - the name of campaign, which leads customers to the advertiser website (non-campaign, Campaign 1, Campaign 2);
- x_7 - the interaction type with the advertising material in media site (non-campaign, impression, click, direct link),
- x_8 - the referrer, e.g. the media name where advertising material was shown for campaign traffic or the external website from where the customer went to the advertiser website (advertiser : frontpage, advertising.com, google.com : AdWords, e-travel.com, msn.com, tradedoubler.com, others);
- x_9 - the advertisement type (non-campaign, flash, search, link, rich media).

Due to large differences in the scales of each numerical feature, the values were normalized so that the values belong to the interval [0, 100].

The numerical data analysis methods were applied to the matrix $\{x_{ji}, j = 0, \dots, m, i = 1, \dots, n\}$ consisting of $m = 2644$ rows and $n = 5$ columns, because this number of rows from 5361 possible ones has no empty cells.

5.3. Methods for analysing and visualizing online shopping behaviour

The results of different visualization methods described in this chapter were published in (Pragarauskaitė, Dzemyda, 2012).

The multidimensional data analysis is a very important task, which could be performed using classification, clustering, or visualizing data. The analysis

results of customer online shopping are usually reviewed and understood by a non-technical person, therefore the results must be displayed in the easiest possible way -visualization of data. Based on the results, different strategies in the advertising campaign could be taken. In this chapter, several direct visualization methods, where each feature is displayed in the visualization, and dimensionality reduction methods, which transform the original data set from R^n to a lower-dimensional space R^d ($d < n$, usually $d = 2$ or 3), are combined in the integrated approach for the visual analysis of customer online shopping data.

The idea of visualization of the multidimensional data will be introduced below in brief. Let the purchase be described by an array of features $x_1, x_2, \dots, x_n$. Any feature may take some numerical values. A combination of values of all the features characterizes a particular purchase $X_j = (x_{j1}, x_{j2}, \dots, x_{jn})$ from the set $\{X_1, X_2, \dots, X_m\}$, where n is the number of features and m is the number of analysed purchases. $X_1, X_2, \dots, X_m$ can be interpreted as the points in the n-dimensional space R^n. In fact, we have a table of numerical data for the analysis: $\{x_{ji}, j = 1, \dots, m, i = 1, \dots, n\}$.

A lot of methods have been developed for the direct data visualization. It is a graphical presentation of the data set providing the quality understanding of the information contents in a natural and direct way: parallel coordinates, scatter plots, survey plots, Chernoff faces, dimensional stacking, etc. (Dzemyda et al., 2012; Grinstein et al., 2001; Hoffman, Grinstein, 2002). There are a lot of so-called projection methods that can be used for reducing the dimensionality, and, particularly, for visualizing the n-dimensional points $X_1, X_2, \dots, X_m \in R^n$. A deep review of the methods is performed e.g. in (Dzemyda et al., 2012; Kaski, 1997; Kohonen, 2001; Kurasova, 2005).

The goal of projection methods is to represent the multidimensional data points in a lower-dimensional space so that certain properties of the structure of the data set were preserved as faithfully as possible. Suppose that we have m data

points, $X_j = (x_{j1}, x_{j2}, \dots, x_{jn}), j = 1, \dots, m$ in the R^n space and, respectively, the goal of dimensionality reduction is to define m points $Y_j = (y_{j1}, y_{j2}, \dots, y_{jd})$, $j = 1, \dots, m$ in the R^d space ($d < n$). The projection can be used to visualize the data set if a sufficiently small output dimensionality is chosen. One of these methods is the principal component analysis (PCA). The well-known principal component analysis (Lawley, Maxwell, 1971) can be used to display the data as a linear projection on a subspace of the original data space such that best preserves the variance in the data. Several approaches have been proposed for reproducing nonlinear multidimensional structures on a lower-dimensional display. The most common methods allocate a representation for each data point in a lower-dimensional space and try to optimize these representations so that the distances between them were as similar as possible to the original distances of the corresponding data items (points). The methods differ in that how the different distances are weighted and how the representations are optimized.

Multidimensional scaling (MDS) refers to a group of methods that is widely used (Borg, Groenen, 1997). The starting data of MDS is a matrix consisting of pair wise dissimilarities of the data items. In general, the dissimilarities need not be distances in the mathematically strict sense. There is a multitude of variants of MDS with slightly different cost functions and optimization algorithms. The MDS algorithms can be roughly divided into two basic types: metric and non-metric MDS. The goal of projection in the metric MDS is to optimize the representations so that the distances between the items in the lower-dimensional space were as close to the original distances as possible. Denote the distance between the points X_i and X_j in R^n by d_{ij}^*, and the distance between the corresponding points Y_i and Y_j in R^d by d_{ij}. In our case, the original dimensionality is n, and the resulting one is $d = 2$. The metric MDS approximates d_{ij}^* by d_{ij}. If a square-error cost is used, the objective function to be minimized can be written as

$$E_{MDS} = \sum_{\substack{i,j=1 \\ i<j}}^{m} w_{ij}\left(d_{ij}^{*} - d_{ij}\right)^2 \qquad (10)$$

where w_{ij} are some positive weights.

The following weights are frequently used:

$$w_{ij} = \left[\sum_{\substack{k,l=1 \\ k<l}}^{m} (d_{kl}^{*})^2\right]^{-1} \qquad (11)$$

$$w_{ij} = \left[d_{ij}^{*} \sum_{\substack{k,l=1 \\ k<l}}^{m} (d_{kl}^{*})^2\right]^{-1} \qquad (12)$$

$$w_{ij} = \frac{1}{m d_{ij}^{*}} \qquad (13)$$

Most often, the Euclidean distances are used for d_{ij} and d_{ij}^{*}. A particular case of the metric MDS is Sammon's mapping (Sammon, 1969). It tries to optimize a cost function called Sammon's stress that describes how well the pairwise distances in a data set are preserved:

$$E_S = \frac{1}{\sum_{\substack{k,l=1 \\ k<l}}^{m} d_{kl}^{*}} \cdot \sum_{\substack{i,j=1 \\ i<j}}^{m} \frac{(d_{ij}^{*} - d_{ij})^2}{d_{ij}^{*}} \qquad (14)$$

Sammon's stress function E_S (14) is more sensitive to small distances than to large ones, i.e. the Sammon's mapping gives more weight to small distances than to large ones.

Artificial neural networks also became as a mean for the dimensionality reduction and data visualization (Dzemyda et al., 2007). The self-organizing map (SOM) (Dzemyda, 2001; Kohonen, 2001) is a class of neural networks that are trained in an unsupervised manner using competitive learning. It is a well-known method for mapping a multidimensional space onto a low-dimensional one. We consider here the mapping onto a two-dimensional grid of neurons. Let $X_1, X_2, \ldots, X_m \in R^n$ be a set of n-dimensional points for mapping. Usually, the neurons are connected to each other via a rectangular or

hexagonal topology. When the training of SOM is completed, the winning neurons are determined for all the multidimensional points. Usually, the grid of neurons is interpreted as a plane and the position of the multidimensional point is completely defined by the position of corresponding wining neuron on the grid.

The integrated approach for the visual analysis of customer online shopping data covers:

- visual analysis by geometric method (scatter plots);
- visual analysis by the multidimensional scaling (Sammon's mapping);
- visual analysis by the artificial neural networks (self organizing map).

5.4. Visual analysis using scatter plots

Due to a large number of parameters and their values of the customer online shopping data, only geometric visualization of scatter plots were used, because the iconographic and hierarchical displays would not be informative for the non-technical person. Scatter plots visualize dependencies between the selected features and allow finding clusters, outliers, and trends within the data. The scatter plot matrix can be used in order to extend the scatter plot to higher dimensions, which is useful for looking at all possible two-way interactions or correlations between features. In Figure 1, the numerical online shopping data is displayed in the scatter plot matrix using RapidMiner software (Rapid Miner software, retrieved 2013) with coloured points by the campaign traffic type (light - campaign, dark - non-campaign).

The advantage of scatter plots is that the visualized features in pairs can be easily interpreted, whereas the main drawback of the scatter plot matrix is that with an increase of the dimensionality, less screen space is available for each projection. In (Colin, Beatty, 1988) the technique to lighten this problem by means of the use of colour is presented, which was also used in Figure 19.

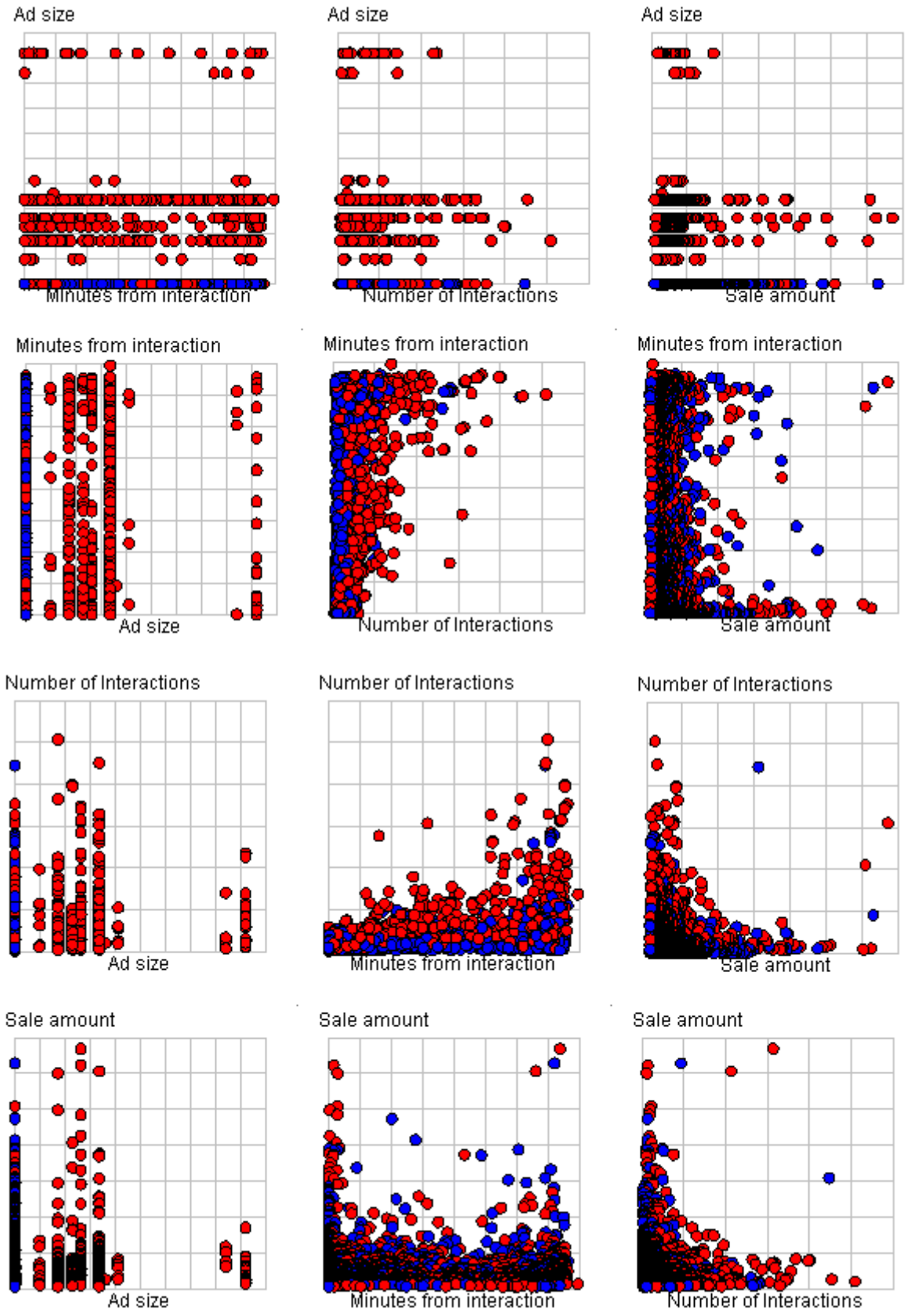


Figure 19. The numerical online shopping data visualization using scatter plots.

Based on the scatter plot visualization, the following conclusions could be drawn:

- Most customers had just few interactions with the campaign advertisements before purchasing anything in the advertiser website.
- Most customers purchased goods in the advertiser website after quite a short period of time since the first interaction.
- The middle size advertisements and non-campaign interactions were the most popular ones in terms of return of investment.
- The customers who bought more expensive goods in the advertiser website belong to both channels: campaign and non-campaign traffic.
- The customers who bought more expensive goods in the advertiser website usually spent either a very short time period or a very long time period since the last interaction compared to the regular customer.
- The customers who bought more expensive goods in the advertiser website had very few interactions before purchasing anything or significantly smaller number of interactions with the campaign material compared to the standard customer.
- There are no separated clusters of customers.

5.5. Visual analysis using the multidimensional scaling (MDS)

For the visualization using multidimensional scaling (MDS), only numerical data was used and it was mapped on the plain ($d = 2$). For the visualization of customer shopping behaviour, Sammon's mapping was used. The visualization results are presented in Figure 20. We do not give legends and units for both axes in Figure 20 with the visualization results, because we are interested in observing the inter-location of the purchases $X_j = (x_{j1}, x_{j2}, \dots, x_{jn}), j = 1, \dots, m$ on a plane only.

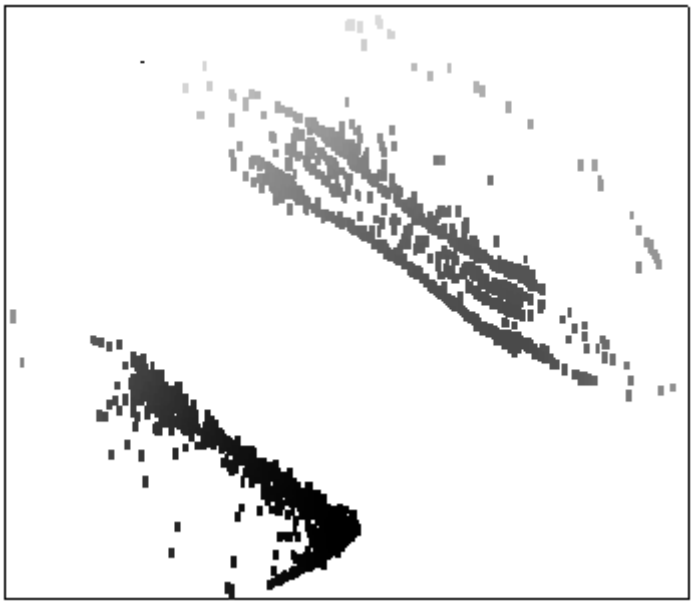


Figure 20. The visualization of numerical online shopping data by Sammon's mapping.

The advantage of MDS visualization is that it shows the relationships between the features of data: multidimensional data points that are similar will appear close together while the points that are different will appear far away from one another. MDS is best used in situations where there is a large amount of data organized in table form. However, particularly for large data sets MDS is slow and time consuming task.

Also, since MDS is a numerical optimization technique, it can fail to find the true best solution because it can become stuck on the local minima, solutions that are not the best solution but that are better than all nearby solutions. The increased computational speed now allows MDS ordinations even of large data sets and multiple features to be run, so that the chance of being stuck on a local minimum is significantly decreased. The experiments were performed with various realizations of MDS in Orange 2.0 tool (Orange 2.0 software, retrieved in 2013). Only Sammon's mapping gave clearly separated clusters, see Figure 19.

Based on the MDS visualization, useful conclusions could be drawn. It is easy to notice separate clusters, which represent different customer groups. If clusters are displayed close to each other in the map, then these clusters have more similarities. The cluster in the bottom of Figure 19 (coloured in black) represents the customers, who came directly to the advertiser website and purchased goods without any interaction with the campaign material. The group of 5 different clusters in the top of Figure 19 represents all the customers who came to the advertiser website through the campaign material. The clusters within this group are divided according to the advertisement size: the top cluster represents the customers who were attracted to the advertiser website via largest advertisements, whereas other clusters are ordered from larger advertisements at the top to smallest advertisements at the bottom. Having the results in Figure 19, the manager can make a decision on the efficiency of the campaign -- how much the size of advertisement influences purchasing.

5.6. Visual analysis using the self organizing maps (SOM)

Viscovery SOMine software (Viscovery Somine, retrieved 2013) was applied to visualize the advertising campaign data using the hexagonal SOM network topology and the Ward neighbourhood function. The classical clustering method of Ward belongs to the hierarchical agglomerative clustering algorithms.

Ward's algorithm uses the following distance between two clusters, A and B:

$$d(A,B) = \sum_{i \in A \cup B} \|X_i - \vec{m}_{A \cup B}\|^2 - \sum_{i \in A} \|X_i - \vec{m}_A\|^2 - \sum_{i \in B} \|X_i - \vec{m}_B\|^2 = \frac{n_A n_B}{n_A + n_B} \|\vec{m}_A - \vec{m}_B\|^2$$

where $\vec{m}_j$ is the centre of cluster j, and n_j is the number of points in it. $d(A, B)$ is called the merging cost of combining the clusters A and B. With hierarchical clustering, the sum of squares starts at zero (because each point is in its own cluster) and then grows as we merge clusters. Ward's algorithm keeps this growth as small as possible.

The SOM network allows analyzing both numerical and categorical data in the easiest and most understandable way for a non-technical person. The disadvantage is that, in order to create a proper SOM visualization, a number of different additional parameters are needed and they should be defined by a person, who already has knowledge about the original data and can prioritize the data in the proper way.

The results on application of the SOM to the online shopping data are discussed below.

Table 8. Cluster information in the SOM visualization of behaviour of Internet users

Cluster	C 1	C 2	C 3	C 4	C 5	C 6	C 7	C 8
Frequency	44.89%	16.15%	13.09%	13.05%	4.88%	1.89%	4.05%	2.00%
Campaign type	0	98.1	100	0	100	100	45.8	73.5
Advertisement size	0	27.3	0	0	35.3	7.9	7.4	58.8
Minutes from last interaction	20.6	36.9	30.1	7.5	43.5	38.2	19.5	32.9
Number of interactions	3.2	12.4	7.3	0.4	15.7	8.8	4.8	10.7
Sale amount	5.6	5.8	6.5	6.9	5.3	8.5	40.7	5.5
Campaign name:								
Non campaign	100%	2%	0%	100%	0%	0%	54%	27%
Campaign 1	0%	98%	100%	0%	100%	100%	46%	8%
Campaign 2	0%	0%	0%	0%	0%	0%	0%	65%
Interaction type:								
Non campaign	100%	1%	0%	0%	0%	0%	44%	0%
Impression	0%	95%	0%	0%	83%	28%	23%	55%
Click	0%	3%	100%	0%	17%	72%	22%	18%
Direct Link	0%	1%	0%	100%	0%	0%	10%	27%
Referrer:								
Advertiser: Frontpage	100%	1%	0%	0%	0%	0%	44%	0%
Advertising.com	0%	98%	0%	0%	0%	0%	24%	0%
Google.com : AdWords	0%	0%	100%	0%	3%	0%	21%	0%
E-travel.com	0%	1%	0%	100%	0%	0%	9%	0%
Msn.com	0%	0%	0%	0%	96%	0%	0%	0%
Tradedoubler.com	0%	0%	0%	0%	0%	100%	0%	0%
Others	0%	0%	0%	0%	1%	0%	2%	100%
Advertisement type:								
Non-campaign	100%	2%	0%	100%	0%	0%	54%	27%
Flash	0%	98%	0%	0%	97%	28%	25%	74%
Search	0%	0%	100%	0%	3%	0%	21%	0%
Link	0%	0%	0%	0%	0%	72%	0%	0%
Rich Media	0%	0%	0%	0%	0%	0%	0%	0%

Eight clusters of SOM neurons were discovered. The 1.5 priority was given to the feature 'sale amount', whereas all the rest features had 1.0 priority in the analysis. The information on each cluster is displayed in Table 8, where the most beneficial customer cluster is C7 and the customers, who belong to this cluster, bought the most expensive goods in the advertiser website. The customers of cluster C7 made around 5 interactions in average with the campaign material before purchasing and the time since the last interaction with the campaign material until the purchase, is around 20 minutes. The customers, who came directly to the advertiser website without interacting with any campaign material (cluster C1), also spent around 20 minutes in the website before purchasing and make 3 interactions in the advertiser website in

average before purchasing. This shows that customers, belonging to both C1 and C7 clusters, have similar behaviour in terms of time spent and interactions made.

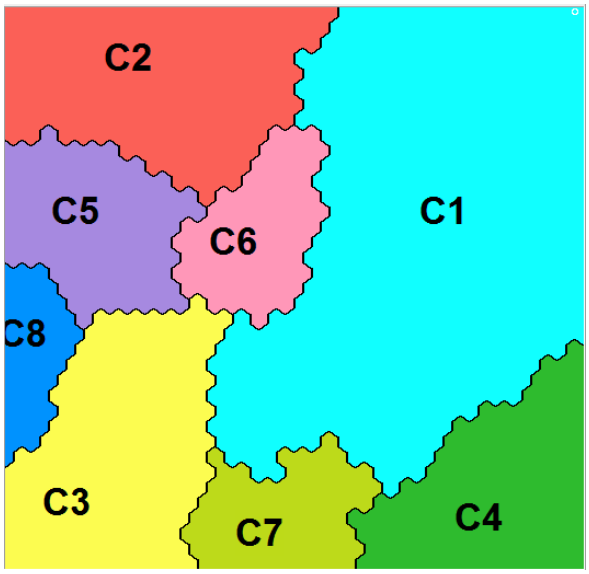


Figure 21. Clusters in SOM obtained by analysing the numerical online shopping data.

The component planes representing numerical features on the SOM are given in Figure 21. The customers that belong to the most beneficial cluster C7 spent the average time period before purchasing, however were not coming back to the advertiser more times than a regular customer. Approximately half of the customers belonging to the C7 cluster came to the advertiser website from the campaign and were interacting with the middle size campaign advertisements.

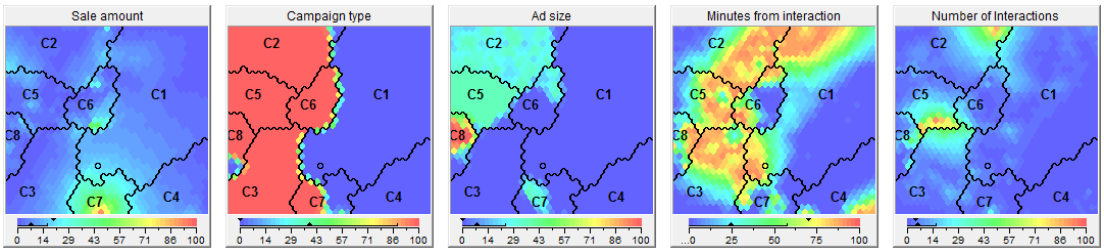


Figure 22. Component planes representing numerical features on the SOM.

The component planes representing categorical features on the SOM are given in Figures 22-25.

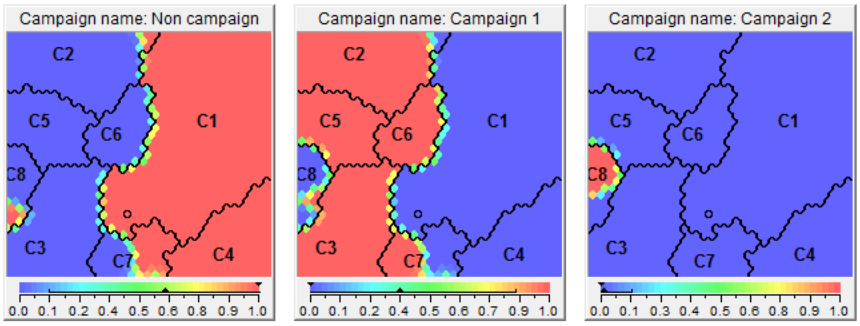


Figure 22. Component planes representing campaign names on the SOM.

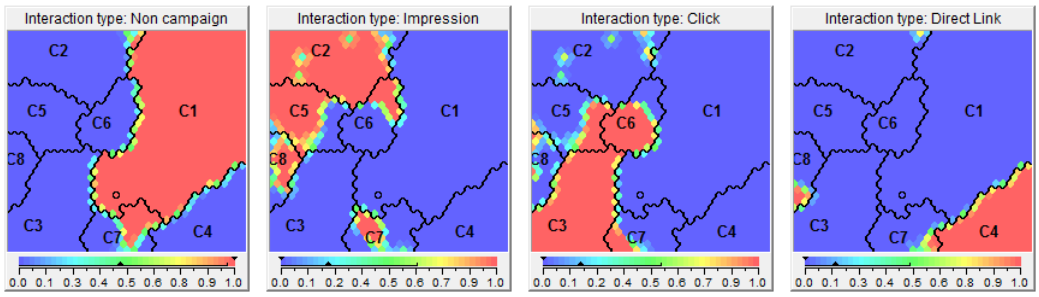


Figure 23. Component planes representing interaction types on the SOM.

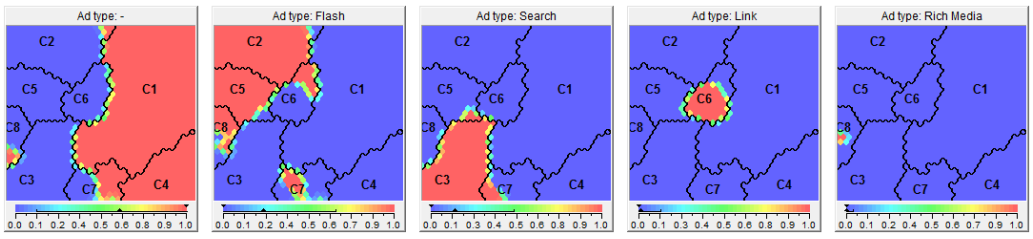


Figure 24. Component planes representing advertisement types on the SOM.

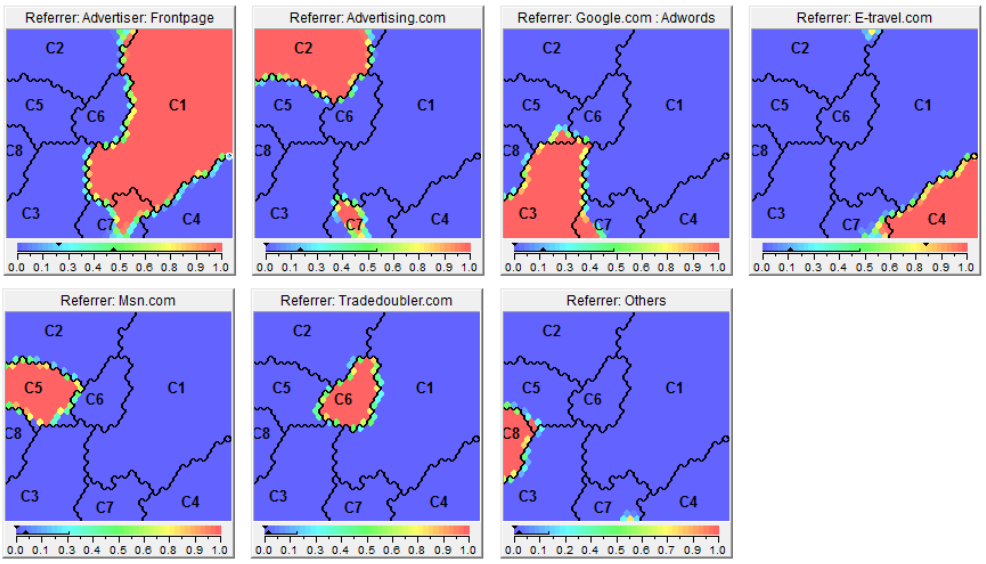


Figure 25. Component planes representing the referrers on the SOM.

The following conclusions could be drawn using the SOM visualization:

- A half of the most beneficial cluster C7 customers came to the advertiser website through Campaign 1 and the other half of C7 customers did not interact with the campaign material;
- The C7 customers who came through the advertising campaign, a half of them saw campaign material and the other half clicked on the campaign advertisements;
- The most popular advertisement type of C7 cluster was flash or search in search engines, whereas the rich media was the least popular advertisement type and does not belong to cluster C7.
- The most beneficial customers from C7 cluster were interacting mostly with advertisements on advertising.com and Google AdWords.

5.7. Conclusions

In this chapter the approach has been proposed for the visual analysis of the online shopping data and their relevance. It integrates several multidimensional data visualization methods of a different nature. The results of the visual analysis of numerical data are combined with the categorical data values.

There is no single visualization that is best for multidimensional data exploration or data mining. Some visualization methods are better for showing clusters or outliers, e.g. self organizing maps or multidimensional scaling, other visualizations can show two way relations, e.g. scatter plots. However, direct visualization methods cannot properly visualize a large number of data records, since the plots become non-informative and difficult to understand. The SOM visualization has showed the best results for the analysis and visualization of the online shopping behaviour; however it requires some training and experience by a user, who will analyze the data. Based on the visualization, the decisions could be taken by a non-technical person who is managing the advertising campaigns. The visualization helps to determine the behaviour of the interesting clusters and their interactions with the campaign material. Thus the decisions on the advertising campaign strategy could be made, e.g. which type of advertisement, on which media sites to put advertisements, etc. Moreover, the decisions on the advertising campaign could be made with a view to increase the return of investment and attract more customers to buy in the online e-shop.

Conclusions

The amount of digital information is increasing by quintillion bytes every day. Big data already reached the level where it becomes difficult to process it using traditional data processing applications. The challenges include the capture of data, storage, search, transfer, analysis, and visualization. On the other hand, such data brings significant benefits and makes it possible to do many things that could not be done previously: predict behaviour of Internet users, diagnose and prevent diseases, identify crime suspects, financial forecasting, etc. Approximate data mining algorithms are very important to efficiently deal with such amount of information and deliver the results within an acceptable timeframe required by various real-world applications.

This thesis focuses on several data mining tasks related to analyzing big data: frequent pattern mining and visual representation of data. For mining frequent patterns in big data, three novel approximate (probabilistic) methods were proposed: RSM (Random Sampling Method), MRM (Multiple Re-sampling Method) and MPBM (Markov Property Based Method). Their performance in terms of the computation speed and precision was evaluated on real and artificial databases and compared with other existing state-of-the-art frequent pattern mining algorithms. For visual representation of big data, the behaviour of Internet users was analyzed using geometric methods, multidimensional scaling and artificial neural networks.

The research completed in this thesis showed the following conclusions:

1) Approximate (probabilistic) frequent sequence mining methods could deliver the acceptable results with right balance between the computation speed and precision when dealing with significantly growing amount of digital information in big data; whereas the exact frequent sequence mining methods work under the time constraints and cannot guarantee the computation speed required by various real-world applications.

2) Novel approximate Random Sampling Method (RSM) brings a significant benefit among existing approximate frequent sequence mining methods by providing a theoretical estimation of error probabilities. It is proved that the error probabilities made by this method could be theoretically estimated using standard statistical methods. The theoretical estimation allows using the proposed method without any extensive empirical experiments, whereas other state-of-the-art approximate methods provide only the empirical evidence based on extensive experiments and observations of algorithm results on different databases.
3) The performance studies demonstrated that a novel approximate Multiple Re-sampling Method (MRM) proposed in this thesis has shown the best overall result in terms of the precision among other approximate algorithms, but it was outperformed by Random Sampling Method (RSM) in terms of the computation speed. If the recommendations on the RSM algorithm parameters are followed, then even more precise results could be achieved.
4) The approximate frequent pattern mining algorithms that require reading the entire original database do not satisfy the time constraints required by many real-world applications. Therefore Markov Property Based Method (MPBM) proposed in this thesis does not satisfy the time constraints as it requires reading the original database at least several times (the number of times equals to the order of the Markov process) and it delivers similar precision results as Multiple Re-sampling Method (MRM).
5) In visual representation of big data, SOM visualization showed the best results for visualizing the behaviour of Internet users; however it requires some training and experience by a user, who is analyzing the data. Based on the visualization, the decisions on the advertising strategy could be taken by a non-technical person who is managing the advertising campaigns.

References

Agrawal, R., Imielinski, T., Swami, A. (1993). Mining association rules between sets of items in large databases. In *ACM SIGMOD Record*, volume 22, ACM, p. 207-216.

Agrawal, R., Srikant, R. (1994). Fast algorithms for mining association rules. In *Proceedings of the 20th International Conference on Very Large Data Bases VLDB*, volume 1215, Citeseer, p. 487-499.

Agrawal, R., Srikant, R. (1995). Mining sequential patterns. In *Proceedings of the Eleventh International Conference of the 11th Int'l Conference on Data Engineering*, IEEE, p. 3-14.

Agarwal, R., Aggarwal, C., Prasad, V. (2001). A tree projection algorithm for generation of frequent itemsets. In *Journal of Parallel and Distributed Computing*, volume 61, issue 3, p. 350–371.

Ayres, J., Flannick, J., Gehrke, J., Yiu, T. (2002). Sequential pattern mining using a bitmap representation. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM, p. 429-435.

Babcock, B., Babu, S., Datar, M., Motwani, R., Widom, J. (2002). Models and Issues in Data Stream Systems. In *21st ACM Symposium on Principles of Database Systems (PODS 2002)*.

Babnik, T., Aggarwal, R. and Moore, P. (2007). Data mining on a transformer partial discharge data using the self-organizing map. IEEE Transactions on Dielectrics and Electrical Insulation, 14 (2), p. 444-452.

Bayardo, R. J. (1998). Efficiently mining long patterns from databases. In SIGMOD 1998, pages 85–93.

Billingsley, P. (1961). Statistical methods in Markov chains, Annals of Mathematical Statistics, v. 32, 1, p. 12-40.

Borg I., Groenen P. (1997). Modern Multidimensional Scaling: Theory and Applications, Springer.

Brin, S., Motwani, R., Ullman, J., Tsur, S. (1997). Dynamic item-set counting and implication rules for market basket data. In *Proceedings of the ACM*

SIGMOD International Conference on Management of Data, volume 26, ACM, p. 255–264.

Bucklin, R. E., Lattin, J. M., Ansari, A., Gupta, S., Bell, D., Coupey, E., Little, J. D. C., Mela, C., Montgomery, A., Steckel, J. (2002). Choice and the Internet: from clickstream to research stream, Marketing Letters, 13(3), p. 245–258.

Bucklin, R. E., Sismeiro, C. (2003). A model of web site browsing behaviour estimated on clickstream data, Journal of Marketing Research, Vol. XL, p. 249–267.

Burdick, D., Calimlim, M., Gehrke, J. (2001). Mafia: A maximal frequent itemset algorithm for transactional databases. In ICDE 2001, Heidelberg, Germany.

Colin, W., Beatty, J. C. (1988). Using color dimensions to display data dimensions, Human Factors, 30(2), p. 127–142.

comScore: The 2010 Europe Digital Year in Review. 2011. Retrieved May 5, 2013, URL: http://www.comscore.com/Insights/Presentations_and_Whitepapers/2011/2010_Europe_Digital_Year_in_Review

Dzemyda, G. (2001). Visualization of a set of parameters characterized by their correlation matrix, Computational Statistics and Data Analysis, 36(1), p. 15–30.

Dzemyda, G., Kurasova, O., Medvedev, V. (2007). Dimension reduction and data visualization using neural networks, In: *Frontiers in Artificial Intelligence and Applications*, Vol. 160, 25–49.

Dzemyda, G., Kurasova, O., Zilinskas, J. (2012). Multidimensional Data Visualization: Methods and Applications, Springer Optimization and Its Applications, Vol. 75, Springer.

DNA Baser software. Retrieved in May 5, 2013. URL: http://www.dnabaser.com/download/download.html.

Feller, W. (1967). An introcution to probability theory and its applications. Vol. I, the 2nd edition. John Willey & Sons, New York.

Francis, Matthew (2012). "Future telescope array drives development of exabyte processing".
Retrieved May 5, 2013, URL: http://arstechnica.com/science/2012/04/future-telescope-array-drives-development-of-exabyte-processing.

Giannella, C., Han, J., Robertson, E., Liu, C. (2003). Mining Frequent Itemsets Over Arbitrary Time Intervals in Data Streams. Technical Report 587, Computer Science Department, Indiana University, retrievedMay 5, 2013, URL: www.cs.indiana.edu/ftp/techreports/index.html.

Goldfarb, A. (2002). Analyzing website choice using clickstream data, In: *Advances in Applied Microeconomic, the Economic of the Internet and E-commerce*, Vol. 11, Elsevier, p. 209–230.

Gouda, K., Hassaan, M., Zaki, M.J. (2007). Prism: A primal-encoding approach for frequent sequence mining. In Data Mining, ICDM 2007. Seventh IEEE International Conference on, p. 487-492.

Gouda, K., Hassaan, M., Zaki, M.J. (2010). Prism: An effective approach for frequent sequence mining via prime-block encoding. In *Journal of Computer and System Sciences*, 76(1), p. 88-102.

Grahne, G., Zhu, J. (2003). Efficiently using prefix-trees in mining frequent itemsets. In: *Proceeding of the ICDM'03 international workshop on frequent itemset mining implementations* (FIMI'03), Melbourne, p. 123–132.

GRC Visitor Browser & Cookie Demographics. Retrieved May 5, 2013, URL: http://www.grc.com/cookies/stats.htm

Grinstein, G., Trutschl, M., Cvek, U. (2001). High-dimensional visualizations, In: *ACM Conference on Knowledge Discovery and Data Mining*, p. 1–14.

Han, J., Pei, J., Yin, Y. (2000). Mining frequent patterns without candidate generation. In ACM SIGMOD Record, volume 29, ACM, p. 1-12.

Han, J., Pei, J., Mortazavi-Asl, B., Chen, Q., Dayal, U., Hsu, M.C. (2000). FreeSpan: Frequent pattern-projected sequential pattern mining. In Proc. Knowledge Discovery and Data Mining, p. 355–359.

Han, J., Pei, J., Mortazavi-Asl, B., Pinto, H., Chen, Q., Dayal, U., Hsu, M.C. (2001). Prefixspan: Mining sequential patterns efficiently by prefix-projected pattern growth. Proceedings of the 17th International Conference on Data Engineering, 215-224.

Han, J. (2002). How Can Data Mining Help BioData Analysis. In BIOKDD02: Workshop on Data Mining in Bioinformatics (with SIGKDD02 Conference).

Han, J., Kamber, M. (2006). Data Mining: Concepts and Techniques, 2nd edition, 2006, Elsevier.

Han, J., Cheng, H., Xin, D., Yan, X. (2007). Frequent pattern mining: current status and future directions. Data Mining and Knowledge Discovery, 15(1), p. 55-86.

Hoffman, P. E., Grinstein, G. G. (2002). A survey of visualizations for high dimensional data mining, In: Information Visualization in Data Mining and Knowledge Discovery, 47–82, Morgan Kaufmann Publishers, San Francisco.

IAB: Consumers Driving the Digital Uptake: The Economic Value of Online Advertising-based Services for Consumers. 2010. Retrieved May 5, 2013, http://www.iabeurope.eu/media/95855/white_paper_consumers_driving_the_digital_uptake.pdf

IBM review: What is big data? — Bringing big data to the enterprise. (2013). Retrieved May 5, 2013. URL: http://www-01.ibm.com/software/data/bigdata.

Kaski, S. (1997). Data Exploration Using Self-Organizing Maps, Acta Polytechnica Scandinavica, Mathematics, Computing and Management in Engineering Series, Vol. 82, Espoo.

Kohonen, T. (2001). Self-Organizing Maps (3rd ed.), Springer Series in Information Sciences 30, Springer.

Kurasova O. Visual Analysis of Multidimensional Data Using Self-Organizing Maps (SOM), Dissertation, Institute of Mathematics and Informatics, Vilnius, 2005.

Kum, H. C., Pei, J., Wang, W., Duncan, D. (2003). ApproxMAP: Approximate Mining of Consensus Sequential Patterns. In Proceedings of the 2003 SIAM International Conference on Data Mining (SIAM DM '03), p. 311–315.

Lawley, D. N., Maxwell, A. E. (1971). Factor Analysis as a Statistical Method (2nd ed.), Butterworths, London.

Liu, G., Lu, H., Lou, W., Yu. J.X. (2003). On computing, storing and querying frequent patterns. In: *Proceeding of the 2003 ACM SIGKDD international conference on knowledge discovery and data mining* (KDD'03), Washington, DC, p. 607–612.

MetaTrader 4 software (Online Trading Platform)- History Center, retrieved 5 May, 2013. URL: http://www.metaquotes.net/data_center.

Moe, W. W. (2003). Buying, searching, or browsing: differentiating between online shoppers using in-store navigational clickstream. In *Journal of Consumer Psychology*, 13, p. 29–40.

Nature journal review: Community cleverness required (2008), Nature 455 (7209), retrieved May 5, 2013. URL: http://www.nature.com/nature/journal/v455/n7209/full/455001a.html.

Norris, J.R. (1997). *Markov Chains*. Cambridge University Press.

Orange 2.0 software. Retrieved May 3, 2011, http://orange.biolab.si/

Park, J. S., Chen, M. S., Yu, P. S. (1995). An effective hash-based algorithm for mining association rules. In *Proceedings of the 1995 ACM SIGMOD international conference on Management of data*, volume 24, ACM, p. 175-186.

Park, J. S., Chen, M. S., Yu, P. S. (1995). Efficient parallel data mining for association rules. In *Proceedings of the fourth international conference on Information and knowledge management*, ACM, p. 31-36.

Park, J., Chen, M., Yu, P. (1997). Using a Hash-based method with transaction trimming for mining association rules," In *IEEE Trans. Knowledge and Data Engineering*, vol. 9, no. 5, p. 813–824.

Park, Y., Fader, P. S. (2004). Modeling browsing behavior at multiple websites, Marketing Science, 23(3), p. 280–303.

Pei, J., Han, J., Mortazavi-Asl, B., Wang, J., Pinto, H., Chen, Q., Dayal, U., Hsu, M.C. (2004). Mining Sequential Patterns by Pattern-growth: The PrefixSpan Approach. In TKDE, Volume 16, Number 11, p. 1424-1440.

Pei, J., Han, J., Mortazavi-Asl, B., Pinto, H., Chen, Q., Dayal, U., Hsu, M.C. (2001). PrefixSpan: mining sequential patterns efficiently by prefix-projected pattern growth. In: *Proceeding of the 2001international conference on data engineering* (ICDE'01), Heidelberg, Germany, p. 215–224.

Pragarauskaitė, J., Dzemyda, G. (2013). Markov Models in the analysis of frequent patterns in financial data. Informatica, Vol. 24, No. 1, p. 87 – 102.

Pragarauskaitė, J., Dzemyda, G. (2011). Probabilistic algorithm for mining frequent sequences. Proceedings ASMDA 2011, Sapienza University of Rome, Edizioni ETS, p. 1454-1460.

Quest Project. IBM Almaden Research Center, San Jose, CA 95120, retried may 5, 2013. URL: http://www.almaden.ibm.com/cs/quest/syndata.html.

Rapid Miner 5.0 software. Retrieved May 5, 2013, http://rapid-i.com.

Sammon J. W. (1969). A nonlinear mapping for data structure analysis, IEEE Transactions on Computers, volume 18, p. 401–409.

Sarawagi, S., Thomas, S., Agrawal, R. (2000). Integrating association rule mining with relational database systems: Alternatives and implications. Data Mining and Knowledge Discovery, 4(2), p. 89-125.

Savasere, A., Omiecinski, E.R., Navathe, S.B. (1995). An efficient algorithm for mining association rules in large databases, Georgia Institute of Technology.

Senecal S., Kalczynski P. J., Nantel J. (2005). Consumers' decision-making process and their online shopping behavior: A clickstream analysis. Journal of Business Research, 58(11), p. 1599–1608.

Silvestri, C., Orlando, S. (2007). Approximate mining of frequent patterns on streams. In Intelligent Data Analysis, Volume 11, Number 1, p. 49-73.

Srikant, R., Agrawal, R. (1995). Mining sequential patterns. In Proceedings ICDE'95. Taipei (Taiwan).

Srikant, R.; Agrawal, R. (1995). Mining Sequential Patterns: Generalizations and Performance Improvements. IBM Almaden Research Center.

Srikant, R., Agrawal, R. (1996). Mining sequential patterns: Generalizations and performance improvements. Advances in Database Technology EDBT'96, p. 1-17.

Srikant, R., Agrawal, R. (1997). Mining generalized association rules. Future Generation Computer Systems, vol. 13, p. 161–180.

Tumasonis, R., Dzemyda, G. (2004). The Probabilistic Algorithm for Mining Frequent Sequences. In *Proceedings ADBIS'04 Eight East-European Conference on Advances in Databases and Information Systems*, p. 89–98.

Toivonen, H. (1996). Sampling large databases for association rules. In *Proceedings of the 22nd VLDB Conference*, p. 134–145.

Usama, F., Piatetsky-Shapiro, G., Smyth, P. (1996). From Data Mining to Knowledge Discovery in Databases. URL: http://www.kdnuggets.com/gpspubs/aimag-kdd-overview-1996-Fayyad.pdf. Retrieved May 5, 2013.

Varadhan, S.R.S. (2001). *Probability theory* (Courrant Lecture Notes), AMS.

Viscovery SOMine 5.2 Software. Retrieved May 5, 2013, URL: http://www.viscovery.net/somine.

Watters, A. (2010). The Age of Exabytes: Tools and Approaches for Managing Big Data. Hewlett-Packard Development Company. Retrieved 2012-10-24.

Webb, G. (2000). Efficient search for association rules. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Boston, MA, p. 99–107.

Yang, Z., Wang, Y., Kitsuregawa, M. (2006). Effective sequential pattern mining algorithms for dense database, in: Japanese Data Engineering Workshop (DEWS).

Zaki, M. J. (2000). Scalable algorithms for association mining. Knowledge and Data Engineering, IEEE Transactions on, 12(3), p. 372-390.

Zaki, M. J. (2001). SPADE: An efficient algorithm for mining frequent sequences. Machine Learning Journal, vol. 42, no. 1–2, p. 31–60.

Zhang, C., Zhang, S., Webb, G.I. (2003). Identifying Approximate Itemsets of Interest in Large Databases. In *Applied Intelligence*, volume 18, p. 91-104.