UNIVERSITÀ DEGLI STUDI DI PADOVA

Head Office: Università degli Studi di Padova
Department: Information Engineering

Ph.D. Course: Information Engineering
Curriculum: Information and Communication Technologies (ICT)
Series: XXXIV, 2022

Rigorous and Scalable Techniques for Mining Patterns from Sequential Data

Coordinator: Prof. Andrea Neviani
Supervisor: Prof. Fabio Vandin

Ph.D. Student: Andrea Tonon

Abstract

Massive amounts of data are generated continuously in every field, such as finance, social networks, medicine, and many others. Data mining is the task of discovering interesting patterns exploring large amounts of data, making those useful and understandable. In recent years, the data mining community has tackled and proposed a vast set of problems but many others are still open. Now more than ever, it is of crucial importance to be able to analyze and extract reliable knowledge from massive datasets. However, this fundamental task poses some challenges. The first one is to design algorithms that scale the computation to the analysis of massive datasets. In such a scenario, very often approaches that return rigorous and high quality approximations are the only viable approach. The second one is to develop strategies that extract useful knowledge providing statistical guarantees on the analysis, while filtering out spurious discoveries. Finally, the abundance of data is opening new scenarios, with a lot of sources generating data continuously, requiring analyses that take into account the sequential nature of the data. The objective of this Thesis is to design novel scalable and rigorous techniques to mine patterns from sequential data, in three scenarios.

The first scenario we consider is mining frequent sequential patterns through sampling. Sequential pattern mining is a fundamental task in data mining and knowledge discovery, with applications in several areas (e.g., biology), that has been extensively studied in the literature, with the definition of several exact methods. However, for large modern sized datasets, the execution of exact methods is computationally very demanding. In such a direction, we develop an algorithm to mine rigorous approximations, defined in terms of false positives or false negatives, of the frequent sequential patterns using sampling. To solve the crucial challenge of computing a sample size which guarantees high quality approximations, we employ the VC-dimension, a key concept from statistical learning theory. Our theoretical and experimental analyses prove that our strategy provides high quality approximations while mining small portions of the original dataset.

The second scenario we consider is mining patterns from samples from unknown probability distributions. In many real life applications (e.g., market basket analysis), the analysis of a dataset is performed to gain insight on the underlying generative process of the data. However, by analyzing only a sample, one cannot exactly solve the problem and has to resort to approximations. In such a scenario, a crucial challenge is to compute a quantity that relates the frequency of a pattern in the sample and its probability of being generated by the underlying process. In this setting, we tackle two problems: the problem of mining, from a single dataset, true frequent sequential patterns, which are sequential patterns frequently generated by the underlying process generating the data; and the problem of mining statistically robust patterns from a sequence of datasets, which are patterns whose probabilities of being generated from the underlying generative processes behind the sequence of datasets follow well specified trends, i.e., increase, decrease or stay stable, through the sequence. For both problems, we develop novel algorithms that return rigorous approximations, defined in terms of false positives or false negatives, employing concepts of statistical learning theory (e.g., the VC-dimension) and we provide theoretical and experimental evidence of their effectiveness.

The last scenario we consider is mining significant patterns. In significant pattern mining, the dataset is seen as a sample from an unknown probability distribution, and the aim is to extract patterns significantly deviating from an assumed null hypothesis with rigorous guarantees in terms of statistical significance of the output, controlling the retrieval of false discoveries. In such a scenario, crucial challenges are the definition of appropriate null models for the data, the assessment of the significance of the patterns of interest, and the multiple hypothesis testing problem. In this setting, we tackle two problems: the problem of mining statistically significant sequential patterns and the problem of mining statistically significant paths in time series data from an unknown network. Since sequential patterns represent ordered sequences of events, significant sequential pattern mining aims to extract sequential patterns that appear more frequently than expected under a null model that randomizes the order of such events inside the sequential patterns. Instead, significant path mining aims to extract patterns that occur more than expected given the distribution of the underlying network that generated the time series, defined as sequences of vertexes of the network. For both problems, we develop novel algorithms that provide rigorous guarantees in term of false discoveries, employing the statistical hypothesis testing framework and techniques based on permutation testing. Our theoretical and experimental analyses provide evidence of their effectiveness.

Acknowledgements

Here I am at the end of this journey. It has been a very challenging path but looking back I can only be happy to have undertaken it, for what I have learned but also for the experiences that I have been able to do during these three years and a half.

First of all I would like to thank my supervisor, Fabio, for all the things I have learned from you, for your patience, and for all your advice. Without you I am sure I would not have taken many steps on this path. In fact, I probably would not have even started it. (Yes, you deserve some other bottles of good wine.)

Then, I would like to thank Mahito and Matthijs for the precious feedback on this Thesis, and Francesco and Geppino for the constructive comments during my year-end presentations.

A big thanks to all the present and past members of our research group, Antonio, Dario, Davide, Diego, Federico, Ilie, Leonardo, and Thach for the many great discussions during our meetings (but also for all the spritz and dinners in Padova).

Impossible not to thank all my friends, for the good times spent together, the laughs and all the adventures faced, hoping there will be many others in the future.

Infine, vorrei ringraziare i miei genitori, Paolo e Maria Rosa, mia sorella Valentina e Andrea, per la pazienza e il supporto che mi avete dedicato in questi anni, non smettendo mai di credere in me. Un ringraziamento speciale anche al piccolo Elia, la cui attesa mi ha tenuto compagnia la notte precedente alla sottomissione di questa Tesi, e a tutti i parenti per l'affetto che sempre mi dimostrate. Ci tengo a ringraziare anche Giuseppe, Mariangela e Lorenzo, per come mi avete accolto in famiglia e per come siete sempre pronti ad ospitarmi. Il ringraziamento finale va a Giulia. Ci siamo conosciuti che ero in procinto di cominciare questo percorso, e in questi tre anni e mezzo non hai mai smesso di rendermi felice con la tua gioia e spensieratezza, nonostante ti sia dovuta sorbire innumerevoli prove di presentazioni in inglese ("Hi everyone, I am Andrea Tonon and in this presentation..." non lo puoi proprio più sentire) e interminabili riletture di paper fino ad orari improbabili le notti che precedevano una submission.

A tutti voi devo tanto e vi ringrazio dal più profondo del mio cuore.

My most sincere gratitude to all of you.

Table of Contents

1 Introduction **1**

2 Background **11**

2.1 Frequent Pattern Mining . . . 11

2.1.1 Itemset Mining . . . 12

2.1.2 Sequential Pattern Mining . . . 13

2.2 Frequent Pattern Mining Through Sampling . . . 14

2.3 True Frequent Pattern Mining . . . 15

2.4 Statistical Learning Theory . . . 16

2.4.1 Maximum Deviation . . . 17

2.4.2 VC-dimension . . . 18

2.5 Significant Pattern Mining . . . 20

2.5.1 Statistical Hypothesis Testing . . . 20

3 Mining Sequential Patterns through Sampling **23**

3.1 Introduction . . . 23

3.1.1 Our Contributions . . . 24

3.1.2 Related Work . . . 25

3.1.3 Organization of the Chapter . . . 26

3.2 Preliminaries . . . 26

3.2.1 Frequent Sequential Pattern Mining . . . 26

3.2.2 Maximum Deviation . . . 28

3.3 VC-Dimension of Sequential Patterns . . . 29

3.3.1 Sample Size for FSP Mining . . . 32

3.4 Sampling-Based Algorithm for FSP Mining . . . 33

3.5 Experimental Evaluation . . . 35

3.5.1 Implementation, Environment, and Datasets . . . 35

3.5.2 Sampling Algorithm Results . . . 37

3.5.3 Analysis of MSNBC 39

4 Mining True Frequent Sequential Patterns 41

4.1 Introduction 41

4.1.1 Our Contributions 42

4.1.2 Related Work 43

4.1.3 Organization of the Chapter 43

4.2 Preliminaries 43

4.2.1 True Frequent Sequential Pattern Mining 44

4.2.2 Maximum Deviation 46

4.3 Upper Bound on the Max Deviation 46

4.4 Algorithm for TFSP Mining 48

4.5 Experimental Evaluation 50

4.5.1 Implementation, Environment, and Datasets 51

4.5.2 True Frequent Sequential Patterns Results 52

5 Mining Statistically Robust Patterns 55

5.1 Introduction 56

5.1.1 Our Contributions 57

5.1.2 Related Work 58

5.1.3 Organization of the Chapter 59

5.2 Preliminaries 59

5.2.1 Pattern Mining 59

5.2.2 Bound on the Maximum Deviation 60

5.3 Statistically Robust Pattern Mining 61

5.4 gRosSo: Approximating the SRPs 63

5.4.1 FPF Approximation of the EP 66

5.4.2 FPF Approximation of the DP 67

5.4.3 FPF Approximation of the SP 67

5.4.4 Guarantees on False Negatives 68

5.4.5 Additional Guarantees of gRosSo 71

5.5 Mining Statistically Robust Sequential Patterns 72

5.5.1 New Upper Bound on the Capacity of a Sequence 73

5.6 Mining Statistically Robust Itemsets 75

5.7 Experimental Evaluation 76

5.7.1 Implementation, Environment, and Datasets 76

5.7.2 Upper Bound on the Capacity 77

5.7.3 Results with Pseudo-Artificial Datasets . . . 79
5.7.4 Results with Real Datasets . . . 84

6 Mining Statistically Significant Sequential Patterns 89
6.1 Introduction . . . 90
6.1.1 Our Contributions . . . 90
6.1.2 Related Work . . . 91
6.1.3 Organization of the Chapter . . . 92
6.2 Preliminaries . . . 93
6.2.1 Significant Sequential Pattern Mining . . . 93
6.2.2 Multiple Hypothesis Testing . . . 95
6.3 PROMISE . . . 96
6.3.1 Efficiently Sampling Random Datasets . . . 98
6.3.2 Parallel Implementation . . . 105
6.4 I-PROMISE . . . 106
6.5 Experimental Evaluation . . . 108
6.5.1 Implementation, Environment, and Datasets . . . 108
6.5.2 Itemsets Swaps Convergence . . . 110
6.5.3 Computational Time of the Generation Strategies . . . 111
6.5.4 Results of PROMISE . . . 112
6.5.5 Results of I-PROMISE . . . 115
6.5.6 Analysis of BIKE with PROMISE . . . 116

7 Mining Statistically Significant Paths 119
7.1 Introduction . . . 120
7.1.1 Our Contributions . . . 121
7.1.2 Related Work . . . 122
7.1.3 Organization of the Chapter . . . 123
7.2 Preliminaries . . . 124
7.2.1 Mining Paths in Time Series Data from a Network . . . 124
7.2.2 k-th Order De Bruijn Graph . . . 125
7.2.3 Multiple Hypothesis Testing for Paths . . . 126
7.3 CASPITA: Mining Statistically Significant Paths . . . 127
7.3.1 Generative Null Model . . . 128
7.3.2 Transactions Oriented Generation (TOG) Strategy . . . 130
7.3.3 Paths Oriented Generation (POG) Strategy . . . 131
7.3.4 Analysis . . . 134

7.3.5 Mining Under-Represented Paths 136
7.3.6 Controlling the Generalized FWER 137
7.4 Mining Statistically Significant Paths from Different Datasets 138
7.5 Experimental Evaluation 139
7.5.1 Implementation, Environment, and Datasets 139
7.5.2 Generation Strategies Comparison 141
7.5.3 Results for POG Strategy with Binomial Approximation 143
7.6 Results with Real Datasets 144
7.6.1 Analysis of BIKE with caSPiTa 147

8 Conclusions **149**

References **153**

Chapter 1

Introduction

Nowadays, massive amount of data is generated every day at an always increasing rate. In the past decades, the generation of this massive amount of data was almost entirely limited to big companies or scientific laboratories, while today even small realities have understood the importance of collecting and analyzing data. Actually, we are all contributing in the generation of new data. How? Sending emails, interacting with our friends on social networks, navigating on the web, even listening to music on our smartphones or watching movies on-demand. We contribute in the generation of data even when we are not online, for example by renting a bike with the bike sharing service of our city or going shopping in a supermarket that is carrying out market analyzes. This Thesis does not want to focus on the dangers and on the ethical aspects that such a collection of data could raise, for example the privacy issues related to the permissions of collecting their data that too often unaware users provide, that of course are crucial problems of our era. Instead, we prefer to optimistically consider the positive aspects of this trend since the abundance of data and the ease with which it can be collected have opened new possibilities and new research directions.

Data mining is the task of discovering interesting patterns exploring large amounts of data, making those useful and understandable. In recent years, the data mining community has developed a vast set of techniques to solve a large set of problems. However, there are still many challenges and open problems. First of all, it is more and more important to be able to *efficiently* analyze such data, since modern sized datasets are continuously growing. In such a direction, to obtain an exact solution is often computationally too expensive and the development of algorithms returning *high quality approximations* is the only viable approach. Secondly, it is of crucial importance to be able to extract *useful* and *reliable knowledge* while filtering out intrinsic noise and uncertainty characteristics of the data. In particular, to be able to assess the *significance* of the results while providing *rigorous guarantees* in term of *spurious discoveries*. Finally, the abundance of data is opening new scenarios. There are a lot of sources that generate data continuously, requiring analyses that take into account the *sequential* nature of the data. *Sequential pattern mining* and *time series analysis* are just two examples of data mining tasks that take into consideration the sequential nature of the data. However, this aspect can be taken into account in almost every data mining task, for example performing analyses on multiple datasets taken in subsequent temporal points.

This Thesis introduces novel *scalable* and *rigorous* techniques to *mine patterns* from *sequential data*. First, we consider the problem of *mining frequent sequential patterns* through *sampling*. Sequential pattern mining is a fundamental task in data mining and knowledge discovery, with applications in several fields, from recommender systems and market basket analysis to biology and medicine, where, in general, sequential patterns describe sequences of events or actions that are useful for predictions in many scenarios. In its original formulation, sequential pattern mining requires to identify all frequent sequential patterns, that is, sequences of itemsets that appear in a sufficient fraction of transactions in a transactional dataset, where each transaction is a sequence of itemsets. Several exact methods have been proposed to find frequent sequential patterns, but the exact solution of the problem requires to process the entire dataset at least once, and often multiple times, making the computation infeasible for large, modern sized datasets. A natural solution to reduce the computation is to use *sampling* to obtain a small random portion (*sample*) of the dataset, and perform the mining process only on the sample. Unfortunately, by analyzing only a sample, the problem cannot be solved exactly, and one has to rely on the approximation provided by the results of the mining task on the sample. Therefore, the main challenge in using sampling is on computing a sample size such that the frequency of the sequential patterns in the sample is close to the frequency that would be obtained from the analysis on the whole dataset. Relating the two quantities using standard techniques (e.g., Hoeffding inequality and union bounds) does not provide adequately small sample sizes, but recently, tools from statistical learning theory (e.g.,Vapnik-Chervonenkis dimension [87] and Rademacher complexity [12]) have been successfully used in frequent itemset mining [66, 67] showing that accurate and rigorous approximations can be obtained from small samples of the entire dataset.

For the task of frequent sequential pattern mining through sampling, this Thesis contributes the following results:

(i) In Chapter 3, we consider the problem of mining *frequent sequential patterns* through *sampling*. In particular, we define two rigorous approximations of the set of frequent sequential patterns: one with no *false negatives* and one with no *false positives*, both defined in terms of a single, easily interpretable, parameter which controls the accuracy of the approximation. We then introduce a new sampling-based algorithm to identify rigorous approximations of the frequent sequential patterns, which do not contain false negatives or false positives with high probability. Our algorithm hinges on a novel bound on the VC-dimension of sequential patterns, and it allows to obtain a rigorous approximation of the frequent sequential patterns by mining only a fraction of the whole dataset. In particular, we provide a simple, but still effective in practice, upper bound on the VC-dimension of sequential patterns by relaxing the upper bound previously defined in [75]. Finally, we perform an extensive experimental evaluation analyzing several datasets. The results show that our algorithm returns high-quality approximations, analyzing only small samples of the original datasets, with guarantees even better than the ones provided by its theoretical analysis. The contributions described in this Chapter appear in [73].

Then, we consider the problem of mining patterns from *samples* drawn from *unknown probability distributions*. In several applications, the analysis of a dataset is performed to gain insight on the *underlying generative process* of the data. For example, in market basket analysis one is interested in gaining knowledge on the behaviour of all customers by analyzing data from a restrict subset of the population. Thus, the task can be modeled as a generative process from which the transactions in the dataset have been drawn. For the problem of mining patterns from samples drawn from unknown probability distributions, in this Thesis, we consider two different scenarios.

The first one is the mining of *true frequent sequential patterns* from a single sample drawn from an unknown probability distribution. In such a scenario, one is *not* interested in sequential patterns that are frequent in the sample, but in sequential patterns that are frequently generated by the *generative process* underlying the data, that is, sequential patterns whose probability of appearing in a transaction generated from the process is sufficiently high. Such patterns, called *true frequent patterns*, have been introduced by [68], which provides a VC-dimension based approach to mine true frequent itemsets. While there is a relation between the probability that a pattern appears in a transaction generated from the process and its frequency in the sample, one cannot simply look at the observed frequency of the patterns. Moreover, due to the stochastic nature of the data, one cannot identify the true frequent patterns with certainty, and approximations are to be sought. In such a scenario, to relate the probability that a pattern appears in a transaction generated from the process with its frequency in the sample is a challenging problem since standard techniques do not provide tight guarantees.

The second one is the mining of *statistically robust patterns* from a *sequence of datasets*. In several real applications, a pattern is studied in the context of a sequence of datasets, where the sequence is given, for example, from the collection of the data at different time points. For example, in market basket analysis, it is natural to study the patterns (e.g., itemsets or sequential patterns) in datasets obtained from transactions in different weeks or months. In almost all applications, one can assume that each dataset is obtained from a *generative process* on transactions, which generates transactions according to a probability distribution. In such a scenario, patterns of interest are the ones whose probability of appearing in a transaction follows some well-specified trends (e.g., it increases, decreases, or is constant across datasets). However, the identification of such patterns is extremely challenging, since the underlying probability distributions are unknown and the observed frequencies of the patterns in the data only approximately reflect such probabilities. As a result, considering the same trends at the level of observed frequencies leads to report several false positives. In addition, techniques developed for significant pattern mining [26] or for statistically emerging pattern mining [38] can only be applied to (a sequence of) two datasets.

For the task of mining patterns from samples drawn from unknown probability distributions, this Thesis contributes the following results:

(i) In Chapter 4, we consider the problem of mining *true frequent sequential patterns*. In particular, we define two rigorous approximations of the set of true frequent sequential patterns: one with no *false negatives* and one with no *false positives*, both defined in terms of a single, easily interpretable, parameter which controls the accuracy of the approximation, as already done for the mining of frequent sequential patterns through sampling. We then introduce a new algorithm to identify rigorous approximations of the true frequent sequential patterns, which do not contain false negatives or false positives with high probability. Our algorithm hinges on a bound on the (empirical) VC-dimension of sequential patterns, as the one proposed in Chapter 3 for the mining of frequent sequential patterns through sampling. Finally, we perform an extensive experimental evaluation analyzing several datasets. The results show that by directly considering the frequency of the patterns in the sample, the output contains false positives and false negatives almost always, while our algorithm returns high-quality approximations with guarantees even better than the ones provided by its theoretical analysis. The contributions described in this Chapter appear in [73].

(ii) In Chapter 5, we consider the problem of mining *statistically robust patterns* from a *sequence of datasets*. In particular, we define the problem of mining statistically robust patterns, and define an approximation of such patterns with no *false positives*. We also describe three general types of patterns, emerging, descending, and stable, which represent different trends of interest in most scenarios. We then introduce an algorithm, gRosSo, to obtain a rigorous approximation of the statistically robust patterns without false positives with high probability. To provide such guarantees, gRosSo can employ any uniform convergence bound. We also define an approximation of the statistically robust patterns with no *false negatives*, and explain how gRosSo can be modified to obtain such an approximation with high probability, besides additional guarantees. We apply the general framework of statistically robust patterns to mine sequential patterns and itemsets. For both pattern mining tasks, we use already-known upper bounds on the empirical VC-dimension of sequential patterns and of itemsets as uniform converge bounds. In particular, for the sequential patterns, we introduce a novel algorithm to compute an upper bound on the capacity of a sequence that can be used in the computation of the bound on the empirical VC-dimension. Finally, we perform an extensive experimental evaluation, mining statistically robust sequential patterns and itemsets from pseudo-artificial datasets, proving that the frequency alone leads to several spurious discoveries, while gRosSo provides high-quality approximations for both data mining tasks. At the end, we analyze real datasets mining statistically robust sequential patterns, proving that gRosSo is able to detect various types of patterns. Part of the contributions described in this Chapter appear in [84], while an extended version is currently under review in the journal Knowledge and Information Systems, invited among the best papers accepted at IEEE ICDM'20.

Finally, we consider the problem of mining *statistically significant patterns*, whose goal is to find patterns *significantly deviating* from an assumed *null hypothesis*, while providing *rigorous guarantees* in term of false positives. For the problem of mining statistically significant patterns, in this Thesis, we consider two different scenarios.

The first one is the mining of *statistically significant sequential patterns*. While the frequency of a pattern is an important feature in some applications, it is usually not sufficient to identify patterns that provide useful knowledge regarding the process described by the data. For example, a sequence of itemsets may appear frequently in a dataset simply because each of its itemsets has high individual frequency, even if there is no real association between them. A natural framework to identify interesting sequential patterns is provided by *statistical hypothesis testing*, where the goal is to mine statistically significant sequential patterns, defined as sequences that appear *more frequently than expected* under an appropriate *null model* for the data. Let us remember that, in general, sequential patterns describe sequences of events or actions. Thus, in the significant sequential pattern mining task, we are interested in mining sequential patterns that appear in a dataset with frequency higher than the one that they would have in random data, obtained randomizing the order of the itemsets but still preserving other characteristics of the original dataset. In such a scenario, to provide guarantees in term of false discoveries is a challenging problem since the gargantuan number of candidate sequential patterns that can be built from a ground set of items poses a severe *multiple hypothesis correction problem.*

The second one is the mining of *statistically significant paths* in *time series data* from an *unknown network.* The mining of time series data has applications in several domains, and in many cases the data are generated by networks, with time series that are sequences of vertexes representing *paths* on such networks. Very often, one has access to a collection of time series, i.e., a collection of sequences of vertexes representing paths on a network, but does not know the distribution on the network that generated them, or the structure of such a network. In such a scenario, we are interested in mining *unexpected* paths from the dataset. Standard techniques usually use the frequency or the number of occurrences as extraction criteria, with the aim to find interesting paths, but, in many real applications, such metrics are not enough to find paths that provide useful knowledge. For example, paths that appear only few times in a dataset may be over-represented if we consider the distribution of the network underlying the data, or vice-versa, paths that appear a lot of times may be under-represented. Thus, techniques based on such metrics may lead to several spurious discoveries. In addition, since we do not know the network underlying the data, we can not directly find over- or under-represented paths.

For the task of mining statistically significant patterns, this Thesis contributes the following results:

(i) In Chapter 6, we consider the problem of mining *statistically significant sequential patterns.* In particular, we introduce a new algorithm, ProMiSe, to identify statistically significant sequential patterns using permutation testing. ProMiSe is the first algorithm to mine statistically significant sequential patterns providing rigorous guarantees on the Family-Wise Error Rate (FWER) of the output, using the Westfall-Young (WY) method to properly correct for multiple hypothesis testing. In such a direction, we introduce and formalize three strategies, based on swaps or on permutations at the level of itemsets, to generate (random) permuted datasets for sequential pattern mining. These three strategies are at the core of ProMiSe, and they sample datasets from the distribution of all datasets where the number of appearances of each itemset and the number of itemsets in each transaction are the same as in the input dataset, but the order of the itemsets is randomized. For the itemsets swapping strategy, we provide a formal analysis proving that a polynomial number of swaps is sufficient to uniformly sample a random dataset from the aforementioned distribution. However, we experimentally show that a number of swaps proportional to the number of itemsets in the dataset is sufficient to sample a random dataset. Then, we introduce an alternative version of ProMiSe, I-ProMiSe, that results in a lower statistical power than the original version but that is several orders of magnitude faster, allowing to mine significant sequential patterns from massive datasets. Finally, we conduct an extensive experimental evaluation with real and pseudo-artificial datasets, providing an implementation of our algorithms. The results show that they allow to efficiently extract significant sequential patterns from real datasets while correctly controlling the FWER. Part of the contributions described in this Chapter appear in [83].

(ii) In Chapter 7, we consider the problem of mining *statistically significant paths* in *time series data* from an *unknown network*. In particular, we introduce the problem of mining statistically significant paths in time series data from an unknown network, defining a generative null model based on meaningful characteristics of the observed dataset. We then introduce caSPiTa, an algorithm to mine statistically significant paths (over- or under-represented) from a time series dataset, while providing guarantees on the probability of reporting at least one false positive, i.e., the FWER, employing the Westfall-Young (WY) method to properly correct for multiple hypothesis testing. caSPiTa requires the generation of random data in accordance with the generative null model, and thus we discuss two possible strategies to generate such data. In addition, for one of the two strategies, we illustrate a possible approximation to assess the significance of the paths more efficiently. Then, we introduce *g*-caSPiTa, a variant of caSPiTa to mine statistically significant paths (over- or under-represented) while providing guarantees on the generalized FWER, which allows to increase the statistical power of the algorithm, with the drawback of tolerating the presence of a few false positives. We also introduce and discuss an alternative interesting scenario in which caSPiTa can be applied, which consists in mining paths that are significant with respect to a null model based on data from a different dataset. Finally, we perform an extensive suite of experiments that demonstrates that caSPiTa is able to efficiently mine statistically significant paths, over- or under-represented, in real datasets while providing guarantees on the false positives. Part of the contributions described in this Chapter appear in [85], while an extended version is currently under review in the journal Knowledge and Information Systems, invited among the best papers accepted at IEEE ICDM'21.

Let us note that each of the problems tackled in this Thesis aims to mine patterns from sequential data in an efficient and rigorous manner taking into consideration different aspects of the data. Thus, each of the proposed techniques allows a different type of data analysis. Here, we briefly recap the setting of each problem, highlighting the key differences between them, to direct interested readers to the most suitable approach according to their requirements. A more in-depth description of the different problem settings is illustrated in the respective Chapters.

In mining frequent sequential patterns through sampling, we consider the classical sequential pattern mining problem, in which one is interested in mining sequential patterns that appear in a sufficiently high number of transactions. Given the massive size of modern datasets, our solution aims to extract frequent sequential patterns by mining only a sample of the input dataset. The output of our algorithm is then a high quality approximation of the frequent sequential patterns that would be mined from the whole dataset, that does not contain false positives or false negatives with high probability, depending on the demands of the user. Thus, this approach allows to efficiently mine frequent sequential patterns from massive datasets obtaining rigorous approximations. It can be used to speed up the execution of traditional frequent sequential pattern mining algorithms for applications in which rigorous approximations are acceptable results or to mine frequent sequential patterns from datasets in which the execution of exact algorithms is infeasible from a computational point of view.

In mining true frequent sequential patterns, we consider a variation of the classical sequential pattern mining problem, in which the dataset is a sample from an unknown probability distribution and one is interested in mining sequential patterns that are frequently generated by the underlying generative process of the dataset. The output of our algorithm is then a rigorous approximation of the sequential patterns that are frequently generated by such a process, that does not contain false positives or false negatives with high probability, depending on the demands of the user, and whose quality depends on the available amount of data. Thus, this approach allows to mine sequential patterns in the scenario in which one has access to a sample from an unknown probability distribution and is interested in gaining knowledge regarding the underlying process by mining sequential patterns.

In mining statistically robust patterns from a sequence of datasets, we consider the scenario in which there is a sequence of datasets that are samples from potentially different unknown probability distributions and one is interested in mining patterns whose probabilities of being generated by such processes follow some well-specified trends. The output of our algorithm is then a rigorous approximation of such patterns, that does not contain false positives or false negatives with high probability, depending on the demands of the user, and whose quality depends on the available amount of data. Thus, this approach allows to mine patterns (e.g., itemsets or sequential patterns) in the scenario in which one has access to a sequence of samples from different unknown probability distributions and is interested in gaining knowledge regarding how the underlying processes evolve.

In mining statistically significant sequential patterns, we consider the scenario in which one is interested in mining sequential patterns that appear more frequently than expected in random data, obtained randomizing the order of the itemsets inside the sequential patterns. The output of our algorithm is then the set of statistically significant sequential patterns, with rigorous guarantees in terms of false positives. Thus, this approach allows to mine sequential patterns whose frequencies are due to the order of their itemsets while filtering out sequential patterns that appear frequent by chance, and then providing sequential patterns that give more information with the drawback of a larger computational cost.

Finally, in mining statistically significant paths in time series data from an unknown network, we consider the scenario in which there is a collection of time series, which are sequences of vertexes representing paths from an unknown network, and one is interested in mining paths that in such a collection appear more ore less than expected given the distribution of the underlying network. The output of our algorithm is then the set of over- or under-represented paths, with rigorous guarantees in terms of false positives. Thus, this approach allows to mine paths in the scenario in which one has access to time series that are data constrained by the structure and by the distribution of a network. A key difference with respect to sequential pattern mining is that in this scenario the paths represent consecutive vertexes in a transaction, while sequential patterns allow gaps between the itemsets in a transaction.

The rest of this Thesis is organized as follows. In Chapter 2, we introduce the notions and main concepts used in the rest of the Thesis, and provide a brief survey of previous works. In Chapter 3, we present our results for the frequent sequential pattern mining through sampling. In Chapter 4, we present our results for the true frequent sequential pattern mining. In Chapter 5, we present our results for the statistically robust pattern mining from a sequence of datasets. In Chapter 6, we present our results for statistically significant sequential pattern mining. In Chapter 7, we present our results for statistically significant path mining in time series from an unknown network. Finally, in Chapter 8, we end this Thesis with some concluding remarks.

Chapter 2

Background

This Thesis introduces novel techniques to mine patterns from sequential data. In this Chapter, we introduce the notions and main concepts used in the rest of the Thesis, and provide a brief survey of previous works. A more in-depth presentation of the previous works related to our novel contributions is illustrated in the respective Chapters. In Section 2.1, we introduce the general framework of frequent pattern mining whose goal is to discover patterns that appear with high frequency over a set of data, and two concrete realizations of such a framework, itemset mining and sequential pattern mining, which represent, respectively, the mining of subsets and sequences of features. In Section 2.2, we introduce the sampling technique in the frequent pattern mining scenario, a solution to reduce the size of the data to analyze. In Section 2.3, we introduce the task of true frequent pattern mining, whose goal is to discover patterns that are frequently generated by the unknown generative process underlying the data. In Section 2.4, we introduce the framework of statistical learning theory and the related concepts of maximum deviation and VC-dimension which may be employed to obtain rigorous guarantees using sampling and in the true frequent pattern mining task. Finally, in Section 2.5, we introduce the task of significant pattern mining, whose goal is to find patterns significantly deviating from an assumed null distribution, and the statistical hypothesis testing framework used to provide rigorous guarantees in the significant pattern mining task.

2.1 Frequent Pattern Mining

Frequent pattern mining is one of the fundamental tasks in data mining and knowledge discovery, and has applications in several domains, ranging form market basket analysis to medicine. In its original formulation, the goal of frequent pattern mining is to discover patterns that appear with high frequency over a set of data. Therefore, such patterns are considered to be interesting as "highly supported" by the data. In particular, the goal of frequent pattern mining is to discover patterns that appear in a fraction at least θ of all the transactions in a transactional dataset, where the threshold θ is a user-specified parameter and its choice must be, at least in part, be informed by domain knowledge.

Let a *dataset* $\mathcal{D} = \{\tau_1, \tau_2, \ldots, \tau_m\}$ be a finite bag of $|\mathcal{D}| = m$ *transactions*, where each transaction is an element from a *domain* $\mathbb{U}$, and let us assume that the elements of $\mathbb{U}$ form a *poset*. We define a *pattern* p as an element of $\mathbb{U}$, potentially with some constraints. For example, in *itemset mining* [2] the domain $\mathbb{U}$ consists of all subsets of binary features called items, while in *sequential pattern mining* [3] it consists in sequences of such subsets. A pattern p *belongs* to a transaction $\tau \in \mathcal{D}$ if and only if p is contained in τ, denoted by $p \sqsubseteq \tau$, where the definition of "belongs" depends on the data mining task. Since the goal of frequent pattern mining is to extract patterns that appear frequently in a set of data, a notion of *frequency* is required. The *support set* $T_{\mathcal{D}}(p)$ of p in $\mathcal{D}$ is the set of transactions in $\mathcal{D}$ containing p

$$T_{\mathcal{D}}(p) = \{\tau \in \mathcal{D} : p \sqsubseteq \tau\}.$$

Finally, the *frequency* $f_{\mathcal{D}}(p)$ of p in $\mathcal{D}$ is the *fraction* of transactions in $\mathcal{D}$ to which p belongs

$$f_{\mathcal{D}}(p) = \frac{|T_{\mathcal{D}}(p)|}{|\mathcal{D}|}.$$

Given a dataset $\mathcal{D}$ and a *minimum frequency threshold* $\theta \in (0, 1]$, *frequent pattern (FP) mining* is the task of reporting the set $FP(\mathcal{D}, \theta)$ of all the patterns whose frequencies in $\mathcal{D}$ are at least θ, and their frequencies, that is,

$$FP(\mathcal{D}, \theta) = \{(p, f_{\mathcal{D}}(p)) : p \in \mathbb{U}, f_{\mathcal{D}}(p) \geq \theta\}.$$

A vast number of different frequent pattern mining tasks has appeared in the literature, considering different types of patterns and different algorithmic strategies to efficiently mine such patterns from large amount of data (see [27] for several references). In the following two Sections, we describe the two pattern mining tasks that we consider in this work, i.e., itemset mining and sequential pattern mining.

2.1.1 Itemset Mining

Itemset mining [2] is probably the most famous pattern mining task and considers the mining of subsets of binary features. Let $\mathcal{I} = \{i_1, i_2, \ldots, i_p\}$ be a finite set of *items*. An *itemset* X is a non-empty subset of $\mathcal{I}$, i.e., $X \subseteq \mathcal{I}$, $X \neq \emptyset$. We denote by $\mathbb{I}$ the set of all possible itemsets composed by items from $\mathcal{I}$. The *length* $|X|$ of X is the number of items in X and an itemset X is contained in an other itemset Y if and only if $X \subseteq Y$.

Example 1. *Let us consider the following dataset* $\mathcal{D} = \{\tau_1, \tau_2, \tau_3, \tau_4\}$ *as an example:*

$$\begin{aligned}\tau_1 &= \{2, 6, 7\}\\ \tau_2 &= \{1, 2, 6, 7\}\\ \tau_3 &= \{1, 2, 3, 4, 5, 6\}\\ \tau_4 &= \{2, 6, 7\}.\end{aligned}$$

The dataset above has 4 transactions. The first one, $\tau_1 = \{2, 6, 7\}$*, has length* $|\tau_1| = 3$*. The frequency* $f_{\mathcal{D}}(\{6, 7\})$ *of the itemset* $\{6, 7\}$ *in* $\mathcal{D}$ *is 3/4, since it is contained in all transactions but* τ_3*.*

2.1.2 Sequential Pattern Mining

Sequential pattern mining [3] is a slightly more recent data mining task where, in general, sequential patterns describe sequences of events or actions that are useful for predictions in many scenarios. Let $\mathcal{I} = \{i_1, i_2, \ldots, i_p\}$ be a finite set of items. Let us remember that an *itemset* X is a non-empty subset of $\mathcal{I}$, i.e., $X \subseteq \mathcal{I}$, $X \neq \emptyset$. A *sequential pattern* (or *sequence*) $s = \langle S_1, S_2, \ldots, S_k\rangle$ is a finite ordered sequence of itemsets, with $S_i \subseteq \mathcal{I}$, $S_i \neq \emptyset$ for all $i \in \{1, \ldots, k\}$. We say that such a sequence s is *built on* $\mathcal{I}$ and we denote by $\mathbb{S}$ the set of all such sequences. The *length* $|s|$ of s is the number of itemsets in s. The *item-length* $||s||$ of s is the sum of the sizes of the itemsets in it, i.e.,

$$||s|| = \sum_{i=1}^{|s|} |S_i|,$$

where the size $|S_i|$ of an itemset S_i is the number of items in it. A sequential pattern $y = \langle Y_1, Y_2, \ldots, Y_a\rangle$ is a *subsequence* of an other sequential pattern $w = \langle W_1, W_2, ..., W_b\rangle$, denoted by $y \sqsubseteq w$, if and only if there exists a sequence of naturals $1 \leq i_1 < i_2 < \cdots < i_a \leq b$ such that $Y_1 \subseteq W_{i_1}, Y_2 \subseteq W_{i_2}, \ldots, Y_a \subseteq W_{i_a}$. Let us note that an item can occur only once in an itemset, but it can occur multiple times in different itemsets of the same sequence. Finally, the *capacity* $c(s)$ of a sequence s is the number of distinct subsequences of s, i.e., $c(s) = |\{a : a \sqsubseteq s\}|$.

Example 2. *Let us consider the following sequential dataset $\mathcal{D} = \{\tau_1, \tau_2, \tau_3, \tau_4\}$ as an example:*

$$\begin{aligned}
\tau_1 &= \langle\{2,6,7\},\{2\}\rangle \\
\tau_2 &= \langle\{1\},\{2\},\{6,7\},\{2\}\rangle \\
\tau_3 &= \langle\{1,4\},\{3\},\{2\},\{1,2,5,6\}\rangle \\
\tau_4 &= \langle\{7\},\{2\},\{6,7\},\{2\}\rangle.
\end{aligned}$$

The dataset above has 4 transactions. The first one, $\tau_1 = \langle\{2,6,7\},\{2\}\rangle$ has length $|\tau_1| = 2$, item-length $||\tau_1|| = 4$ and capacity $c(\tau_1) = 14$. The frequency $f_{\mathcal{D}}(\langle\{7\},\{2\}\rangle)$ of the sequence $\langle\{7\},\{2\}\rangle$ in $\mathcal{D}$, is 3/4, since it is contained in all transactions but τ_3. Let us note that the sequence $\langle\{7\},\{2\}\rangle$ occurs three times as a subsequence of τ_4, but τ_4 contributes only once to the frequency of $\langle\{7\},\{2\}\rangle$.

2.2 Frequent Pattern Mining Through Sampling

Several exact methods have been proposed to find frequent patterns [27], but the exact solution of the problem requires processing the entire dataset typically several times, which may be infeasible for large modern sized datasets. A natural solution to reduce the computation is to use *sampling*, in order to obtain a small random portion of the dataset, a *sample*, and to perform the mining process only on the sample [82, 61]. We define a sample $\mathcal{S} = \{s_1, s_2, \dots, s_n\}$ of a dataset $\mathcal{D}$ as a bag of $|\mathcal{S}| = n$ transactions, where each transaction is an element from $\mathcal{D}$ sampled *with replacement*. It is easy to see that by analyzing only a sample of the data, the problem cannot be solved exactly, and one has to rely on the approximation provided by the results of the mining task on the sample. Therefore, the main challenge in using sampling is on computing a sample size such that the frequency of the patterns in the sample is close to the frequency that would be obtained from the analysis on the whole dataset in order to obtain rigorous guarantees on the reported patterns. In particular, usually, one is interested in mining sets of patterns without *false positives*, i.e., $FP(\mathcal{S},\theta) \subseteq FP(\mathcal{D},\theta)$, or without *false negatives*, i.e., $FP(\mathcal{D},\theta) \subseteq FP(\mathcal{S},\theta)$. The problem is that to relate the two quantities, i.e., the frequency of the patterns in the sample and the frequency of the patterns in whole dataset, using standard techniques (e.g., Hoeffding inequality and union bounds) does not provide sample sizes adequately small. In fact, such procedures require the knowledge of the number of all the patterns in the dataset, which is impractical to compute in a reasonable time. Thus, one has to resort to loose upper bounds that usually result in sample sizes that are larger than the whole dataset. Recently, tools from statistical learning theory (see Section 2.4), such as Vapnik-Chervonenkis dimension [87] and Rademacher complexity [12], have been successfully used in frequent itemset mining [66, 67], showing that accurate and rigorous approximations can be obtained from small samples of the entire dataset. Related to sampling, the VC-dimension has also been used to approximate frequent substrings in collections of strings [57], while the related concept of pseudo-dimension has been used to mine interesting subgroups [69].

2.3 True Frequent Pattern Mining

Besides the classical frequent pattern mining task (see Section 2.1), a vast number of variations of this problem has been proposed in the literature, considering alternative concepts to the frequency to mine the patterns of interest. Examples are significant pattern mining [26], high-utility pattern mining [19, 20], and true frequent pattern mining [68]. In particular, [68] introduces the task of mining *true frequent itemsets*, which are itemsets that are frequently generated by the unknown generative process underlying the data.

In several applications, the dataset $\mathcal{D}$ is a sample of transactions independently drawn from an unknown probability distribution π on $\mathbb{U}$, that is, the dataset $\mathcal{D}$ is a finite bag of $|\mathcal{D}|$ *independent identically distributed* (i.i.d.) samples from π, with $\pi : \mathbb{U} \to [0,1]$. The *true support set* $T(p)$ of p is the set of patterns in $\mathbb{U}$ to which p belongs, i.e., $T(p) = \{\tau \in \mathbb{U} : p \sqsubseteq \tau\}$, and the *true frequency* $t_\pi(p)$ of p with respect to (w.r.t.) π is the probability that a transaction sampled from π contains p, that is,

$$t_\pi(p) = \Pr_{\tau \sim \pi}(p \sqsubseteq \tau).$$

In such a scenario, the final goal of the data mining process on $\mathcal{D}$ is to gain a better understanding of the process that generated the data, i.e., the distribution π, through the true frequencies of the patterns, which are unknown and only approximately reflected in the dataset $\mathcal{D}$. Thus, given a probability distribution π on $\mathbb{U}$ and a minimum frequency threshold $\theta \in (0,1]$, *true frequent pattern (TFP) mining* is the task of reporting the set $TFP(\pi,\theta)$ of all patterns whose true frequencies w.r.t. π are at least θ, and their true frequencies, that is,

$$TFP(\pi,\theta) = \{(p, t_\pi(p)) : p \in \mathbb{U}, t_\pi(p) \geq \theta\}.$$

A crucial problem of this task is that it is not possible to find the exact set $TFP(\pi, \theta)$ given a finite number of random samples from π, the dataset $\mathcal{D}$. Thus, one has to resort to approximations of $TFP(\pi, \theta)$ to obtain sets of patterns without *false positives*, i.e., $FP(\mathcal{D}, \theta) \subseteq TFP(\pi, \theta)$, or without *false negatives*, i.e., $TFP(\pi, \theta) \subseteq FP(\mathcal{D}, \theta)$. The major challenge to obtain such approximations is in relating the true frequency of the patterns w.r.t. the unknown probability distribution with the frequency of the patterns in the dataset. In particular, [68] employs the empirical VC-dimension of itemsets, a concept of statistical learning theory, to relate such two quantities and they find rigorous approximations which do not contain false positives with high probability. However, their solution requires to solve an optimization problem that is tailored to itemsets and whose computation could be expensive.

2.4 Statistical Learning Theory

Sampling is a powerful technique at the core of statistical data analysis and machine learning that attempts to estimate properties of an entire domain using a finite, often small, set of observations from such a domain. Using sampling, a crucial challenge is to understand the *sample complexity* of the problem, i.e., the minimum sample size needed to obtain the required results with rigorous guarantees [52]. *Statistical learning theory* [86] is an important branch of machine learning and pattern recognition that aims to provide quantitative probabilistic guarantees on the performances of learning algorithms. Related to sampling, statistical learning theory provides concepts to control the sample complexity of the problem, in order to obtain rigorous guarantees on the quality of the results.

In the task of frequent pattern mining through sampling, one is interested in generating a small portion of the dataset, a sample, and to mine the patterns from the sample, in order to speed up the computation of the mining phase or to obtain a sample that can fit in the main memory of the machine used for the mining. In particular, one needs to compute a sample size such that the frequency of the patterns in the sample is close to the frequency that would be obtained from the analysis on the whole dataset, in order to obtain rigorous guarantees on the patterns reported from the sample. Instead, in the task of true frequent pattern mining, one is interested in mining patterns that are frequently generated by the unknown generative process underlying the data. The major challenge is that the underlying probability distribution is unknown and thus, to obtain rigorous approximations, one needs to relate the true frequency of the patterns w.r.t. the unknown probability distribution and the frequency of the patterns in the dataset. Even if the two tasks seem to be unrelated, they both require to relate a quantity and its *empirical average* on a sample: the frequency of a pattern in a dataset and its frequency in a sample, for the frequent pattern mining through sampling task, or the true frequency of a pattern w.r.t. an unknown probability distribution and its frequency in a sample from such a distribution, for the true frequent pattern mining task. To relate such quantities, concepts from statistical learning theory can be used. In the next two Sections, we introduce the concept of maximum deviation, which formally defines the quantity that we need to bound in order to relate the quantities that we have just described, and the VC-dimension, a concept from statistical learning theory that can be used to compute a probabilistic upper bound on the maximum deviation.

2.4.1 Maximum Deviation

Let $\mathbb{X}$ be a domain set, let π be a probability distribution on $\mathbb{X}$, i.e., $\pi : \mathbb{X} \to [0, 1]$, and let $\mathcal{G}$ be a set of functions from $\mathbb{X}$ to $[0, 1]$. Given a function $g \in \mathcal{G}$, we denote by $\mathbb{E}[g]$ the *expectation* of g, that is,

$$\mathbb{E}[g] = \mathbb{E}_{x\sim\pi}[g(x)],$$

with $x \in \mathbb{X}$, while, given a sample $\mathcal{A}$ of $|\mathcal{A}|$ elements drawn from $\mathbb{X}$ w.r.t. π, the *empirical average* $E(g, \mathcal{A})$ of g on $\mathcal{A}$ is defined as

$$E(g, \mathcal{A}) = \frac{1}{|\mathcal{A}|} \sum_{x_i\in\mathcal{A}} g(x_i).$$

The *maximum deviation* $D(\mathcal{G}, \mathcal{A})$ is defined as the largest absolute difference, over all functions $g \in \mathcal{G}$, between the expectation of g and its empirical average on a sample $\mathcal{A}$, that is,

$$D(\mathcal{G}, \mathcal{A}) = \sup_{g\in\mathcal{G}} |\mathbb{E}[g] - E(g, \mathcal{A})|. \tag{2.1}$$

To have a bound $\mu \in (0, 1)$ on the maximum deviation, i.e., $D(\mathcal{G}, \mathcal{A}) \leq \mu$, it is known as *uniform convergence* and implies that all estimates $E(g, \mathcal{A})$ are uniformly close to (i.e., within a factor μ) their true values $\mathbb{E}[g]$. Or equivalently, it implies simultaneous bounds on the expected values of the functions $g \in \mathcal{G}$ from their estimates.

In the frequent pattern mining through sampling task, one is interested in finding good estimates for the frequencies of the patterns in the dataset, simultaneously for all the patterns. In such a scenario, the frequency of a pattern in the dataset and its frequency on the sample represent, respectively, the expectation and the empirical average of a function associated with such a pattern. Instead, in the true frequent pattern mining task, one is interested in finding good estimates for the true frequencies of the patterns, simultaneously for all the patterns. In such a scenario, the true frequency of a pattern and its observed frequency in the dataset represent, respectively, the expectation and the empirical average of a function associated with such a pattern. Thus, in both scenarios, tools from statistical learning theory, e.g., VC-dimension [87, 52] and Rademacher complexity [76, 12], can be used to find probabilistic upper bounds on the maximum deviation, i.e., finding a $\mu \in (0,1)$ such that

$$\Pr\left(D(\mathcal{G},\mathcal{A}) \leq \mu\right) \geq 1-\delta,$$

with a confidence parameter $\delta \in (0,1)$. More common techniques (e.g., Hoeffding inequality and union bounds) instead do not provide useful results since they require to know the number of all possible patterns to consider, which can be infinite or impractical to compute.

2.4.2 VC-dimension

The Vapnik-Chervonenkis (VC) dimension [87, 52] of a space of points is a measure of the complexity or expressiveness of a family of indicator functions, or, equivalently, of a family of subsets, defined on that space. A finite bound on the VC-dimension of a structure implies a bound of the number of random samples required to approximately learn that structure.

We define a *range space* as a pair $(X,\mathcal{R})$, where X is a finite or infinite set and $\mathcal{R}$, the *range set*, is a finite or infinite family of subsets of X. The members of X are called *points*, while the members of $\mathcal{R}$ are called *ranges*. Given $A \subseteq X$, we define the *projection* of $\mathcal{R}$ in A as $P_{\mathcal{R}}(A) = \{r \cap A \ : \ r \in \mathcal{R}\}$. We define 2^A as the *power set* of A, that is the set of all the possible subsets of A, including the empty set $\emptyset$ and A itself. If $P_{\mathcal{R}}(A) = 2^A$, then A is said to be *shattered by* $\mathcal{R}$. The VC-dimension of a range space is the cardinality of the largest set shattered by the space.

Definition 1. *Let $RS = (X,\mathcal{R})$ be a range space and $B \subseteq X$. The empirical VC-dimension $EVC(RS,B)$ of RS on B is the maximum cardinality of a subset of B shattered by $\mathcal{R}$. The VC-dimension $VC(RS)$ of RS is defined as $VC(RS) = EVC(RS,X)$.*

Example 3. *Let $X = [0,1]$ be the set of all the points in $[0,1]$ and let $\mathcal{R}$ be the set of subsets $[a,b]$, with $0 \leq a \leq b \leq 1$, that is $[a,b] \subseteq [0,1]$. Let us consider the set $Y = \{x,y,z\}$, containing 3 points $0 \leq x < y < z \leq 1$. It is not possible to find a range whose intersection with the set Y is $\{x,z\}$, since all the ranges $[a,b]$, with $0 \leq a \leq b \leq 1$,* *containing x and z, also contain y. Then, $VC(X, \mathcal{R})$ must be less than 3. Let us consider now the set $Y = \{x, y\}$, containing only 2 points $0 \leq x < y \leq 1$. It is easy to see that Y is shattered by $\mathcal{R}$, then $VC(X, \mathcal{R}) = 2$.*

The main application of VC-dimension in statistics and learning theory is to derive the sample size needed to approximately "learn" the ranges, as defined below.

Definition 2. *Let $RS = (X, \mathcal{R})$ be a range space and let γ be a probability distribution on X. Given $\mu \in (0, 1)$, a bag B of elements taken from X in accordance with γ is a μ-bag of X w.r.t. γ if for all $r \in \mathcal{R}$,*

$$\left| \Pr_{\gamma}(r) - \frac{|B \cap r|}{|B|} \right| \leq \mu.$$

A μ-bag of X w.r.t. γ can be constructed sampling points from X according to the distribution γ, as follows.

Theorem 1 ([43]). *Let $RS = (X, \mathcal{R})$ be a range space of VC-dimension $VC(RS) \leq d$, and let γ be a probability distribution on X. Given $\mu, \delta \in (0, 1)$, there is a constant $c > 0$ such that if B is a bag of $|B| = m$ elements sampled from X in accordance with γ, where*

$$m \geq \frac{c}{\mu^2}\left(d + \ln \frac{1}{\delta}\right),$$

then B is a μ-bag of X w.r.t. γ with probability $\geq 1 - \delta$.

The universal constant c has been experimentally estimated to be at most 0.5 [46], and in the remaining of this Thesis, we will use $c = 0.5$. Let us note that Theorem 1 holds also when d is an upper bound on the empirical VC-dimension $EVC(RS, B)$ of RS on B [43]. In that case, the bag B itself is a μ-bag of X w.r.t. γ. How to compute an upper bound on the (empirical) VC-dimension depends on the pattern mining task and efficiently computable upper bounds on the VC-dimension of sequential patterns and itemset will be illustrated in the following Chapters.

In what follows, we may omit the probability distribution γ when it represents the uniform distribution. In particular, when γ is the uniform distribution and the set X is a finite set, a bag B of elements taken from X is a μ-bag of X (see Definition 2) if for all $r \in \mathcal{R}$,

$$\left| \frac{|X \cap r|}{|X|} - \frac{|B \cap r|}{|B|} \right| \leq \mu. \tag{2.2}$$

2.5 Significant Pattern Mining

As already stated above, besides the classical frequent pattern mining task, a vast number of variations of this problem has been proposed in the literature, considering alternative concepts to the frequency to mine the patterns of interest. In *significant pattern mining* [90] the dataset is seen as a sample from an unknown distribution and one is interested in finding patterns *significantly deviating* from an assumed *null distribution* (or hypothesis). Many variants and algorithms have been proposed for the problem [26, 35, 60, 59]. In the significant pattern mining task, a crucial challenge is to be able to correctly identify significant patterns obtaining a limited number of false discoveries. To obtain guarantees on the number of false discoveries, the *statistical hypothesis testing framework* is used.

2.5.1 Statistical Hypothesis Testing

The task of mining *statistically significant patterns* is to identify patters significantly deviating from an assumed *null hypothesis* H_0, which represents the case “nothing interesting” for a given question of interest. In a simple statistical test, the null hypothesis H_0 is tested against an *alternative hypothesis* H_1. Given a dataset $\mathcal{D}$ and a pattern w, let $a_{\mathcal{D}}(w)$ be the *test statistic* of the pattern w computed in the dataset $\mathcal{D}$, which is a quantity of interest in the given question for the pattern w in $\mathcal{D}$. In particular, the test statistic quantifies an aspect of w that would distinguish the null from the alternative hypothesis. For example, in many statistically significant pattern mining tasks, one is interested in finding patterns w that appear in a dataset $\mathcal{D}$ with a frequency higher of the one that they have under an appropriate null distribution. In such a scenario, the test statistic $a_{\mathcal{D}}(w)$ of w computed in $\mathcal{D}$ is the frequency $f_{\mathcal{D}}(w)$ of w in $\mathcal{D}$. Under the null hypothesis, the test statistic of w is described by a random variable A_w, and in order to assess the significance of w, a p-value p_w is commonly computed. The p-value p_w of w is the probability of observing an outcome of the test statistic A_w, under the null hypothesis H_0, that is equally or more extreme than the test statistic $a_{\mathcal{D}}(w)$ of w in $\mathcal{D}$, that is,

$$p_w = \Pr\left[A_w \text{ “equally or more extreme than” } a_{\mathcal{D}}(w) | H_0\right].$$

The set of outcomes that should be considered “equally or more extreme than” depends on the goal of the test, and it will be defined in the respective Chapters.

Unfortunately, for complex null hypothesis, the p-values can not be computed analytically, since there is not a closed form for A_w. However, when one can generate data from the distribution described by the null hypothesis, the p-values can be estimated by a simple Monte Carlo (MC) procedure as follows: to generate M datasets $\tilde{\mathcal{D}}_i$, with $i \in \{1, \ldots, M\}$, from the distribution described by the null hypothesis. Then, the p-value p_w is estimated as

$$p_w = \frac{1}{M+1}\left(1 + \sum_{i=1}^{M} \mathbb{1}\left[a_{\tilde{\mathcal{D}}_i}(w) \text{ ``equally or more extreme than'' } a_{\mathcal{D}}(w)\right]\right),$$

where $\mathbb{1}[\cdot]$ is the indicator function of value 1 if the argument is true, and 0 otherwise.

The statistical hypothesis testing framework is commonly used to provide guarantees on the false discoveries, i.e., patters flagged as significant while they are not. When a single pattern w is tested for significance, flagging w as significant, i.e., rejecting the null hypothesis, when $p_w \leq \alpha$, where $\alpha \in (0,1)$ is the *significance threshold* fixed by the user, guarantees that the probability that w corresponds to a false discovery $\leq \alpha$.

The situation is completely different when several patters are tested simultaneously, as in the case of significant pattern mining. If d patterns are tested with the approach used for a single pattern, i.e., each pattern is flagged as significant if its p-value is $\leq \alpha$, then the expected number of false discoveries can be as large as αd. To solve this issue, one identifies a *corrected significance threshold* $\delta \in (0,1)$ such that all patterns with p-value $\leq \delta$ can be reported as significant while providing some guarantees on the number of false discoveries. A common approach is to identify δ that provides guarantees on the *Family-Wise Error Rate (FWER)*, defined as the probability of reporting at least one false positive, that is, if FP is the number of false positives, then FWER $=$ Pr[FP $>$ 0]. For a given value δ, let FWER(δ) be the FWER obtained when δ is used as corrected significance threshold, that is, by reporting as significant all patterns with p-value $\leq \delta$. Often FWER(δ) can not be evaluated in closed form, and thus approaches, as the Bonferroni correction or based on permutation testing, described below, must be employed.

A simple approach to correct for multiple hypothesis testing is to use the *Bonferroni correction* [11], setting $\delta = \alpha/d$. Using the union bound, it is easy to show that the resulting FWER satisfies FWER(δ) $\leq d\delta = \alpha$. However, to properly correct for multiple hypothesis testing, one must consider the number of all patterns that can be generated by the distribution described by the null hypothesis, and when such a number d is very large, as in the case of pattern mining, δ is very close to 0, resulting in low *statistical power* and many *false negatives*, i.e., significant patterns that are not correctly reported in output [89–91].

The *Westfall-Young (WY) method* [95] is a multiple hypothesis testing procedure based on *permutation testing* that results in high statistical power and that has been successfully applied in pattern mining scenarios [81, 79, 45, 60]. The WY method directly estimates the joint distribution of null hypotheses using permuted datasets, i.e., datasets obtained from the distribution described by the null hypothesis. In detail, the WY method considers P datasets $\tilde{\mathcal{D}}_i$, with $i \in \{1, \ldots, P\}$, generated from the distribution described by the null hypothesis. Then, for every dataset $\tilde{\mathcal{D}}_i$, with $i \in \{1, \ldots, P\}$, it computes the minimum p-value $p_{min}^{(i)}$ over all patterns of interest in $\tilde{\mathcal{D}}_i$. The FWER FWER(δ) obtained using δ as corrected significance threshold can then be estimated as

$$\mathrm{FWER}(\delta) = \frac{1}{P} \sum_{i=1}^{P} \mathbb{1} \left[p_{min}^{(i)} \le \delta \right]. \tag{2.3}$$

Thus, given a *FWER threshold* $\alpha \in (0, 1)$, the corrected significance threshold δ^* is obtained as

$$\delta^* = \max\{\delta : \mathrm{FWER}(\delta) \le \alpha\}.$$

Chapter 3

Mining Sequential Patterns through Sampling

In this Chapter, we apply sampling to a classical pattern mining task, frequent sequential pattern mining. In such a scenario, we aim to perform the mining of frequent sequential patterns in a small random sample of the original dataset with the aim to speed up the execution by reducing the quantity of data to analyze. Using sampling, a key challenge is the computation of a sample size which guarantees that the frequencies of the sequential patterns in the dataset are close to their frequencies in the sample. To solve this difficult task, we use the statistical learning theory concept of VC-dimension, which allows us to obtain rigorous approximations of the results that would have been obtained by analyzing the whole dataset, with guarantees on the false positives or on the false negatives. The contributions described in this Chapter appear in [73].

3.1 Introduction

Let us remember, from Section 2.1.2, that sequential pattern mining [3] is a fundamental task in data mining and knowledge discovery, with applications in several fields, from recommender systems and e-commerce to biology and medicine, where, in general, sequential patterns describe sequences of events or actions that are useful for predictions in many scenarios. In its original formulation, sequential pattern mining requires to identify all *frequent sequential patterns*, that is, sequences of itemsets that appear in a fraction at least θ of all the transactions in a transactional dataset, where each transaction is a sequence of itemsets. The threshold θ is a user-specified parameter and its choice must be, at least in part, be informed by domain knowledge.

Several exact methods have been proposed to find frequent sequential patterns (e.g., [28, 54, 64, 96]). However, the exact solution of the problem requires processing the entire dataset at least once, and often multiple times. For large, modern sized datasets, this may be infeasible. A natural solution to reduce the computation is to use *sampling* (see Section 2.2) to obtain a small random portion, i.e., a *sample*, of the dataset, and perform the mining process only on the sample. It is easy to see that by analyzing only a sample of the data the problem cannot be solved exactly, and one has to rely on the approximation provided by the results of the mining task on the sample. Therefore, the main challenge in using sampling is on computing a sample size such that the frequency of the sequential patterns in the sample is close to the frequency that would be obtained from the analysis on the whole dataset. Relating the two quantities using standard techniques (e.g., Hoeffding inequality and union bounds) does not provide useful results, that is, small sample sizes. In fact, such procedures require the knowledge of the number of all the sequential patterns in the dataset, which is impractical to compute in a reasonable time. So, one has to resort to loose upper bounds that usually result in sample sizes that are larger than the whole dataset. Recently, tools from statistical learning theory (see Section 2.4), e.g.,Vapnik-Chervonenkis dimension [87] and Rademacher complexity [12], have been successfully used in frequent itemsets mining [66, 67], a frequent pattern mining task where transactions are collections of items, showing that accurate and rigorous approximations can be obtained from small samples of the entire dataset. While sampling has previously been used in the context of sequential pattern mining (e.g., [65]), to the best of our knowledge no sampling algorithm providing a rigorous approximation of the frequent sequential patterns has been proposed.

3.1.1 Our Contributions

In this Chapter, we study the problem of mining *frequent sequential patterns* through *sampling* and we propose an efficient algorithm for this problem, based on the concept of VC-dimension. In this regard, our contributions are:

- We define two rigorous approximations of the set of frequent sequential patterns: one with no *false negatives*, that is, containing all elements of the original set; and one with no *false positives*, that is, without any element that is not in the original set. Our approximations are defined in terms of a single parameter, which controls the accuracy of the approximation and is easily interpretable.

- We study the VC-dimension of sequential patterns, an advanced concept from statistical learning theory, and provide a simple, but still effective in practice, upper bound on the VC-dimension of sequential patterns by relaxing the upper bound previously defined in [75].

- We introduce a new sampling-based algorithm to identify rigorous approximations of the frequent sequential patterns with probability $1 - \delta$, where δ is a confidence parameter set by the user. Our algorithm hinges on our novel bound on the VC-dimension of sequential patterns, and it allows to obtain a rigorous approximation of the frequent sequential patterns by mining only a fraction of the whole dataset.
- We perform an extensive experimental evaluation analyzing several sequential datasets, showing that our algorithm provide high-quality approximations, even better than guaranteed by its theoretical analysis.

3.1.2 Related Work

Since the introduction of the frequent sequential pattern mining problem [3], a number of exact algorithms has been proposed for this task, ranging from multi-pass algorithms using the anti-monotonicity property of the frequency function [77], to prefix-based approaches [54], to works focusing on the closed frequent sequences [88].

The usage of sampling to reduce the amount of data for the mining process while obtaining rigorous approximations of the collection of interesting patterns has been successfully applied in many pattern mining tasks. Raïssi and Poncelet [65] provided a theoretical bound on the sample size for a single sequential pattern in a static dataset using Hoeffding concentration inequalities, and they introduced a sampling approach to build a dynamic sample in a streaming scenario using a biased reservoir sampling. Our work is heavily inspired by the work of Riondato and Upfal [66], which introduced an advanced statistical learning technique for the task of frequent itemset and association rules mining. In particular, they employed the concept of VC-dimension to derive a bound on the sample size needed to obtain an approximation of the frequent itemsets and association rules from a dataset. The VC-dimension has also been used to approximate frequent substrings in collections of strings [57] and for approximate mining frequent subgraphs [63], while the related concept of pseudo-dimension has been used to mine interesting subgroups [69].

Other works have studied the problem of approximating frequent sequential patterns using approaches other than sampling. In [49], the dataset is processed in blocks with a streaming algorithm, but the intermediate sequential patterns returned may miss many frequent sequential patterns. More recently, [75] introduced an algorithm to process the datasets in blocks using a variable, data-dependent frequency threshold, based on an upper bound on the empirical VC-dimension, to mine each block. They define an approximation for frequent sequential patterns that is one of the definitions we consider in this Chapter. The intermediate results obtained after analyzing each block have probabilistic approximation guarantees, and after analyzing all blocks the output is the exact collection of frequent sequential patterns. While these works, in particular [75], are related to our contributions, they do not provide sampling algorithms for sequential pattern mining.

3.1.3 Organization of the Chapter

The rest of the Chapter is structured as follows. Section 3.2 contains the main definitions and concepts used throughout this Chapter. In Section 3.3, we apply the statistical learning theory concept of VC-dimension to sequential patterns and introduce a strategy to compute a sample size which guarantees to obtain a probabilistic upper bound on the maximum deviation for the frequent sequential pattern mining through sampling task. In Section 3.4, we introduce our sampling based algorithm to mine a rigorous approximation of the frequent sequential patterns. Finally, Section 3.5 reports the results of an extensive suite of experiments performed to evaluate the effectiveness and performance of our sampling strategy.

3.2 Preliminaries

We now define the definitions and concepts used throughout the Chapter, refreshing some notions defined in Chapter 2. We start by refreshing the task of sequential pattern mining, introduced in Section 2.1.2, and formally define the problem of approximating the frequent sequential patterns. We then apply the concept of maximum deviation, introduced in Section 2.4.1, to the frequent pattern mining scenario.

3.2.1 Frequent Sequential Pattern Mining

Let us remember, from Section 2.1.2, that a *sequential pattern*, or *sequence*, $s = \langle S_1, S_2, \ldots, S_k \rangle$ is a finite ordered sequence of *itemsets* S_i, with $i \in \{1, \ldots, k\}$, which are sets of elements called *items*. The *length* $|s|$ of s is the number of itemsets in s, the *item-length* $||s||$ of s is the sum of the sizes of the itemsets in s, that is,

$$||s|| = \sum_{i=1}^{|s|} |S_i|,$$

while the *capacity* $c(s)$ of a sequence s is the number of distinct subsequences of s, i.e., $c(s) = |\{a : a \sqsubseteq s\}|$.

Let the *domain* $\mathbb{S}$ be the set of all the sequences which can be built with itemsets that are subsets of a given set of items $\mathcal{I}$. A *dataset* $\mathcal{D}$ is a finite bag of $|\mathcal{D}|$ *(sequential) transactions* where each transaction is a sequence from $\mathbb{S}$. A sequence s *belongs* to a transaction $\tau \in \mathcal{D}$ if and only if s is a subsequence of τ, i.e., $s \sqsubseteq \tau$. For any sequence s, the *support set* $T_{\mathcal{D}}(s)$ of s in $\mathcal{D}$ is the set of transactions in $\mathcal{D}$ to which s belongs, that is, $T_{\mathcal{D}}(s) = \{\tau \in D : s \sqsubseteq \tau\}$. Finally, the *frequency* $f_{\mathcal{D}}(s)$ of s in $\mathcal{D}$ is the *fraction* of transactions in $\mathcal{D}$ to which s belongs, that is,

$$f_{\mathcal{D}}(s) = \frac{|T_{\mathcal{D}}(s)|}{|\mathcal{D}|}.$$

Given a dataset $\mathcal{D}$ and a *minimum frequency threshold* $\theta \in (0, 1]$, *frequent sequential pattern* (FSP) mining is the task of reporting the set $FSP(\mathcal{D}, \theta)$ of all the sequences whose frequency in $\mathcal{D}$ is at least θ, and their frequencies, that is,

$$FSP(\mathcal{D}, \theta) = \{(s, f_{\mathcal{D}}(s)) : s \in \mathbb{S}, f_{\mathcal{D}}(s) \geq \theta\}.$$

In this Chapter, we are interested in finding the set $FSP(\mathcal{D}, \theta)$ by mining only a sample of the dataset $\mathcal{D}$. Let us note that given a sample of the dataset $\mathcal{D}$, one cannot guarantee to find the exact set $FSP(\mathcal{D}, \theta)$ and has to resort to approximations of $FSP(\mathcal{D}, \theta)$. Thus, we are interested in finding rigorous approximations of $FSP(\mathcal{D}, \theta)$. In particular, we consider the approximation of $FSP(\mathcal{D}, \theta)$ defined in [75].

Definition 3. *Given $\varepsilon \in (0, 1)$, a false negatives free (FNF) ε-approximation $\mathcal{N}$ of $FSP(\mathcal{D}, \theta)$ is defined as a set of pairs (s, f_s),*

$$\mathcal{N} = \{(s, f_s) : s \in \mathbb{S}, f_s \in [0, 1]\},$$

that has the following properties:

- *$\mathcal{N}$ contains a pair (s, f_s) for every $(s, f_{\mathcal{D}}(s)) \in FSP(\mathcal{D}, \theta)$;*
- *$\mathcal{N}$ contains no pair (s, f_s) such that $f_{\mathcal{D}}(s) < \theta - \varepsilon$;*
- *for every $(s, f_s) \in \mathcal{N}$, it holds $|f_{\mathcal{D}}(s) - f_s| \leq \varepsilon/2$.*

(Let us note that while [75] introduced the definition of FNF ε-approximation of $FSP(\mathcal{D}, \theta)$, it did not provide a sampling algorithm to find such an approximation for a given $\varepsilon \in (0, 1)$.)

Intuitively, the approximation $\mathcal{N}$ contains all the frequent sequential patterns that are in $FSP(\mathcal{D},\theta)$ (i.e., there are no *false negatives*) and no sequential pattern that has frequency in $\mathcal{D}$ much below θ. In addition, $\mathcal{N}$ provides a good approximation of the actual frequency of the sequential patterns in $\mathcal{D}$, within an error $\varepsilon/2$, arbitrarily small.

Depending on the application, one may be interested in a different approximation of $FSP(\mathcal{D},\theta)$, where all the sequential patterns in the approximation are frequent sequential patterns in the whole dataset.

Definition 4. *Given $\varepsilon \in (0,1)$, a false positives free (FPF) ε-approximation $\mathcal{F}$ of $FSP(\mathcal{D},\theta)$ is defined as a set of pairs (s, f_s),*

$$\mathcal{F} = \{(s, f_s) : s \in \mathbb{S}, f_s \in [0,1]\},$$

that has the following properties:

- *$\mathcal{F}$ contains no pair (s, f_s) such that $f_{\mathcal{D}}(s) < \theta$;*
- *$\mathcal{F}$ contains all the pairs (s, f_s) such that $f_{\mathcal{D}}(s) \geq \theta + \varepsilon$;*
- *for every $(s, f_s) \in \mathcal{F}$, it holds $|f_{\mathcal{D}}(s) - f_s| \leq \varepsilon/2$.*

The approximation $\mathcal{F}$ does not contain *false positives*, that is, sequences with $f_{\mathcal{D}}(s) < \theta$. In addition, it does not miss sequences with $f_{\mathcal{D}}(s) \geq \theta + \varepsilon$ and, similarly to the FNF ε-approximation, we have that, for every pair in $\mathcal{F}$, it gives a good approximation of the actual frequency of the sequential patterns in $\mathcal{D}$, within an error $\varepsilon/2$, arbitrarily small.

3.2.2 Maximum Deviation

In the frequent pattern mining through sampling task (see Section 2.2), we aim to find good estimates for $f_{\mathcal{D}}(s)$ simultaneously for all the sequential patterns $s \in \mathbb{S}$. In such a scenario, the frequency $f_{\mathcal{D}}(s)$ is the expectation of a Bernoulli random variable (r.v.) $X_{\mathcal{D}}(s,\tau)$ which is 1 if the sequential pattern s appears in a transaction τ drawn uniformly at random from $\mathcal{D}$, that is,

$$\mathbb{E}_{\tau\sim\mathcal{D}}[X_{\mathcal{D}}(s,\tau)] = \Pr_{\tau\sim\mathcal{D}}(X_{\mathcal{D}}(s,\tau) = 1) = \frac{|T_{\mathcal{D}}(s)|}{|\mathcal{D}|} = f_{\mathcal{D}}(s).$$

Let $\mathcal{S}$ be a sample of transactions drawn uniformly and independently at random from $\mathcal{D}$. We define the frequency $f_{\mathcal{S}}(s)$ as the fraction of transactions of $\mathcal{S}$ where s appears. In this scenario, we have that the frequency $f_{\mathcal{D}}(s)$ of s on $\mathcal{D}$ and the frequency $f_{\mathcal{S}}(s)$ of s

on the sample $\mathcal{S}$ represent, respectively, the expectation $\mathbb{E}(g_s)$ and the empirical average $E(g_s, \mathcal{S})$ of a function g_s associated with a sequential pattern s. Thus, in such a scenario, the maximum deviation (see Equation 2.1) is

$$\sup_{s \in \mathbb{S}} |f_{\mathcal{D}}(s) - f_{\mathcal{S}}(s)|.$$

In the next Section, we provide a strategy, based on the concept of VC-dimension of sequential patterns, to compute a sample size which guarantees to obtain a probabilistic upper bound on the maximum deviation in the frequent sequential pattern mining scenario.

3.3 VC-Dimension of Sequential Patterns

In this Section, we apply the statistical learning theory concept of VC-dimension (see Section 2.4.2) to sequential patterns. First, we define the range space associated with a sequential dataset. Then, we show a computable efficient upper bound on the VC-dimension and we show how to compute the size of a sample that guarantees to obtain a good approximation for the problem of mining the frequent sequential patterns.

Let us remember that a range space is a pair $(X, \mathcal{R})$ where X contains points and $\mathcal{R}$ contains ranges. For a sequential dataset, X is the dataset itself, while $\mathcal{R}$ contains the sequential transactions that are the support set for some sequential patterns.

Definition 5. *Let $\mathcal{D}$ be a sequential dataset consisting of sequential transactions, which are sequences from a domain $\mathbb{S}$. We define $RS = (X, \mathcal{R})$ to be a range space associated with $\mathcal{D}$ such that:*

- *$X = \mathcal{D}$ is the set of sequential transactions in the dataset;*
- *$\mathcal{R} = \{T_{\mathcal{D}}(s) : s \in \mathbb{S}\}$ is a family of sets of sequential transactions such that for each sequential pattern s, the set $T_{\mathcal{D}}(s) = \{\tau \in \mathcal{D} : s \sqsubseteq \tau\}$ is the support set of s on $\mathcal{D}$.*

The VC-dimension of this range space is the maximum size of a set of sequential transactions that can be shattered by the support sets of the sequential patterns.

Example 4. *Let us consider the following dataset $\mathcal{D} = \{\tau_1, \tau_2, \tau_3, \tau_4\}$ as an example:*

$$\tau_1 = \langle \{1\}, \{2, 3\}, \{4, 5, 6\} \rangle$$
$$\tau_2 = \langle \{1\}, \{3\}, \{4\} \rangle$$
$$\tau_3 = \langle \{7\}, \{3, 4\} \rangle$$
$$\tau_4 = \langle \{4\}, \{5\} \rangle.$$

The dataset above has 4 transactions. We now show that the VC-dimension of the range space $RS = (X, \mathcal{R})$ associated with $\mathcal{D}$ (see Definition 5) is 2. Let us consider the set $A = \{\tau_2, \tau_3\}$. The power set 2^A of A is $2^A = \{\emptyset, \{\tau_2\}, \{\tau_3\}, \{\tau_2, \tau_3\}\}$. A is shattered by $\mathcal{R}$ since the projection $P_{\mathcal{R}}(A)$ of $\mathcal{R}$ in A is equal to 2^A (let us remember that $P_{\mathcal{R}}(A) = \{r \cap A \ : \ r \in \mathcal{R}\}$):

$$\begin{aligned}
\emptyset &= A \cap T_{\mathcal{D}}(\langle\{6\}\rangle),\\
\{\tau_2\} &= A \cap T_{\mathcal{D}}(\langle\{1\}\rangle),\\
\{\tau_3\} &= A \cap T_{\mathcal{D}}(\langle\{3,4\}\rangle),\\
A = \{\tau_2, \tau_3\} &= A \cap T_{\mathcal{D}}(\langle\{3\}\rangle).
\end{aligned}$$

Since $|A| = 2$ and A is shattered by $\mathcal{R}$, then the range space associated with $\mathcal{D}$ has VC-dimension ≥ 2. Analogously, the sets $\{\tau_1, \tau_3\}$,$\{\tau_1, \tau_4\}$,$\{\tau_2, \tau_4\}$ and $\{\tau_3, \tau_4\}$ are shattered by $\mathcal{R}$. The set $B = \{\tau_1, \tau_2\}$ is instead not shattered by $\mathcal{R}$: since $\tau_2 \sqsubseteq \tau_1$, there is not a sequential pattern $\tilde{s}$ such that $B \cap T_{\mathcal{D}}(\tilde{s}) = \{\tau_2\}$. The sets $C = \{\tau_1, \tau_3, \tau_4\}$ and $E = \{\tau_2, \tau_3, \tau_4\}$ are not shattered by $\mathcal{R}$ either: there is not a sequential pattern $\hat{s}$ such that $\{\tau_3, \tau_4\} = C \cap T_{\mathcal{D}}(\hat{s})$ or $\{\tau_3, \tau_4\} = E \cap T_{\mathcal{D}}(\hat{s})$. Thus, the VC-dimension of the range space associated with $\mathcal{D}$ is exactly 2.

The exact computation of the VC-dimension of the range space associated with a dataset $\mathcal{D}$ is computationally expensive. The s-index, introduced by Servan-Schreiber et al. [75], provides an efficiently computable upper bound on the VC-dimension of sequential patterns. Such an upper bound is based on the notion of capacity $c(s)$ of a sequence s. Let us remember that the capacity $c(s)$ of a sequence s is the number of distinct subsequences of s, that is, $c(s) = |\{a : a \sqsubseteq s\}|$. The exact capacity can be computed using the algorithm described in [16], but it is computationally expensive and may be prohibitive for large datasets. Instead, [75] proposed an algorithm to compute a more efficient upper bound $\tilde{c}(s) \geq c(s)$. Let us consider that a first naïve bound is given by $2^{||s||} - 1$, that may be a loose upper bound of $c(s)$ because it is obtained by considering all the items contained in all the itemsets in s as distinct, that is, the capacity of the sequence s is $2^{||s||} - 1$ if and only if all the items contained in all the itemsets of the sequence s are different. The bound proposed by [75] can be computed as follows. When s contains, among others, two itemsets A and B such that $A \subseteq B$, subsequences of the form $\langle C \rangle$ with $C \subseteq A$ are considered twice in $2^{||s||} - 1$, "generated" once from A and once from B. To avoid over-counting such $2^{|A|} - 1$ subsequences, [75] proposes to consider only the ones "generated" from the longest itemset that can generate them. Then, the s-index is defined as follows.

Definition 6 ([75])**.** *Let $\mathcal{D}$ be a sequential dataset. The* s-index *of $\mathcal{D}$ is the maximum integer d such that $\mathcal{D}$ contains at least d different sequential transactions with upper bound on their capacities $\tilde{c}(s)$ at least $2^d - 1$, such that no one of them is a subsequence of another, that is the d sequential transactions form an anti-chain.*

The following result from [75] shows that the s-index is an upper bound on the VC-dimension of the range space for sequential patterns in $\mathcal{D}$.

Theorem 2 (Lemma 3 [75])**.** *Let $\mathcal{D}$ be a sequential dataset with s-index d. Then, the range space $RS = (X, \mathcal{R})$ corresponding to $\mathcal{D}$ has VC-dimension $\leq d$.*

While an upper bound on the s-index can be computed in a streaming fashion, it still requires to check whether a transaction is a subsequence of one of the transactions that define the current value of the s-index. In addition, the computation of the upper bound $\tilde{c}(s)$ on the capacity of a sequence s requires to check whether the itemsets of s are subsets of each others. In a scenario in which the efficiency is of key importance, such as sampling, these operations may be prohibitive from a computational point of view. Thus, to avoid such expensive operations, we define an upper bound on the s-index, which we call *s-bound*, that does not require to check whether the transactions form an anti-chain, nor the computation of an upper bound on the capacity of a sequence.

Definition 7. *Let $\mathcal{D}$ be a sequential dataset. The* s-bound *of $\mathcal{D}$ is the maximum integer d such that $\mathcal{D}$ contains at least d different sequential transactions with item-length at least d.*

In practice, it is quite uncommon that the long sequences that define the value of the s-index are subsequences of other sequences, thus, removing the anti-chain constraint, the bound does not deteriorate. In addition, the usage of the naïve algorithm to compute the upper bound on $c(s)$, that is $2^{||s||} - 1$, it is equivalent to consider the transactions that have item-length at least d to calculate the s-bound, making the computation much faster without worsening the bound on the VC-dimension in practice.

Algorithm 1 shows the pseudo-code to compute an upper bound on the s-bound in a streaming fashion. It uses an ordered set to maintain in memory the set of transactions that define the current value of the s-bound. The ordered set stores pairs $(\tau, ||\tau||)$ composed by a transaction τ and its item-length $||\tau||$, sorted by decreasing item-length. In addition, it uses an hash set to speed up the control on the equal transactions.

Algorithm 1: `SBoundUpp`: compute an upper bound on the s-bound.

```
Data: Dataset $\mathcal{D}$.
Result: Upper bound $d$ on the $s$-bound of $\mathcal{D}$.
1  $\mathcal{H} \leftarrow$ empty HashSet of transactions;
2  $\mathcal{O} \leftarrow$ empty OrderedSet of pairs $(\tau, ||\tau||)$ sorted by decreasing item-length $||\tau||$;
3  $d \leftarrow 0$;
4  foreach $\tau \in \mathcal{D}$ do
5      if $\tau \notin \mathcal{H}$ then
6          $\ell \leftarrow$ ComputeItemLength($\tau$);
7          if $\ell > d$ then
8              $\mathcal{H}$.add($\tau$);
9              $\mathcal{O}$.add(($\tau, \ell$));
10             ($\tau', \ell'$) $\leftarrow \mathcal{O}$.last();
11             if $\ell' > d$ then $d \leftarrow d + 1$;
12             else
13                 $\mathcal{H}$.remove($\tau'$);
14                 $\mathcal{O}$.removeLast();
15 return $d$;
```

3.3.1 Sample Size for FSP Mining

In this Section, we show how to compute a sample size m for a random sample $\mathcal{S}$ of transactions taken from $\mathcal{D}$ such that the maximum deviation is bounded by $\varepsilon/2$, that is, $\sup_{s\in\mathbb{S}} |f_{\mathcal{D}}(s) - f_{\mathcal{S}}(s)| \leq \varepsilon/2$, for a user-defined value $\varepsilon \in (0, 1)$, using an upper bound on the VC-dimension of sequential patterns, e.g., the s-bound (see Definition 7). Such a result underlies the sampling algorithm that will be introduced in Section 3.4.

Theorem 3. *Let $\mathcal{S}$ be a random sample of m transactions taken with replacement from the sequential dataset $\mathcal{D}$ and let d be an upper bound on the VC-dimension of the range space associated with $\mathcal{D}$. Given $\varepsilon, \delta \in (0, 1)$, if*

$$m \geq \frac{2}{\varepsilon^2}\left(d + \ln\frac{1}{\delta}\right),$$

then $\sup_{s\in\mathbb{S}} |f_{\mathcal{D}}(s) - f_{\mathcal{S}}(s)| \leq \varepsilon/2$ *with probability at least* $1 - \delta$.

Proof. From Theorem 1, we know that $\mathcal{S}$ is an $\varepsilon/2$-bag for $\mathcal{D}$ with probability at least $1 - \delta$. Then, from Definition 2 (and Equation 2.2),

$$\left|\frac{|\mathcal{D} \cap r|}{|\mathcal{D}|} - \frac{|\mathcal{S} \cap r|}{|\mathcal{S}|}\right| \leq \frac{\varepsilon}{2}$$

for all $r \in \mathcal{R}$. Given a sequence $s \in \mathbb{S}$ and its support set $T_{\mathcal{D}}(s)$ on $\mathcal{D}$, that is the range r_s, and from the definition of range set of a sequential dataset, Definition 5, we have

$$\frac{|\mathcal{D} \cap r_s|}{|\mathcal{D}|} = f_{\mathcal{D}}(s)$$

and

$$\frac{|\mathcal{S} \cap r_s|}{|\mathcal{S}|} = f_{\mathcal{S}}(s).$$

Thus, $\sup_{s\in\mathbb{S}} |f_{\mathcal{D}}(s) - f_{\mathcal{S}}(s)| \leq \varepsilon/2$ with probability $\geq 1-\delta$, which concludes the proof. □

Algorithm 2: `ComputeSampleSize`: compute the sample size.

```
    Data: Dataset $\mathcal{D}$, $\varepsilon, \delta \in (0,1)$.
    Result: Sample size $m$ s.t. $\Pr(\sup_{s\in\mathbb{S}} |f_{\mathcal{D}}(s) - f_{\mathcal{S}}(s)| \leq \varepsilon/2) \geq 1-\delta$.
1   $d \leftarrow$ SBoundUpp($\mathcal{D}$);
2   $m \leftarrow 2/\varepsilon^2 (d + \ln(1/\delta))$;
3   return $m$;
```

Algorithm 2, using an efficient upper bound on the VC-dimension of sequential patterns, i.e., the s-bound (described in Algorithm 1), shows how to compute a sample size which guarantees that $\sup_{s\in\mathbb{S}} |f_{\mathcal{D}}(s) - f_{\mathcal{S}}(s)| \leq \varepsilon/2$ with probability $\geq 1-\delta$. This algorithm is used in our sampling strategy (Algorithm 3 of Section 3.4).

3.4 Sampling-Based Algorithm for FSP Mining

We now present a sampling algorithm for frequent sequential pattern mining. The aim of this algorithm is to reduce the amount of data to consider to mine the frequent sequential patterns, in order to speed up the extraction of the sequential patterns and to reduce the amount of memory required. We define a *random sample* as a bag of m transactions taken uniformly and independently at random, with replacement, from $\mathcal{D}$. Obtaining the exact set $FSP(\mathcal{D},\theta)$ from a random sample is not possible, thus we focus on obtaining an ε-approximation with probability at least $1-\delta$, where $\delta \in (0,1)$ is a *confidence* parameter, whose value, with $\varepsilon \in (0,1)$, is provided in input by the user. Intuitively, if a random sample is sufficiently large, then the set of frequent sequential patterns extracted from the random sample well approximates the set $FSP(\mathcal{D},\theta)$. The challenge is to find the number of transactions that are necessary to obtain the desired ε-approximation, FNF or FPF. To compute such a sample size, our approach uses the VC-dimension of sequential patterns (see Section 3.3.1).

Theorem 4. *Given $\varepsilon, \delta \in (0,1)$, let $\mathcal{S}$ be a random sample of size m sequential transactions taken independently at random with replacement from the dataset $\mathcal{D}$ such that $\sup_{s\in\mathbb{S}} |f_{\mathcal{D}}(s) - f_{\mathcal{S}}(s)| \leq \varepsilon/2$ with probability at least $1-\delta$. Then, given $\theta \in (0,1]$, the set $FSP(\mathcal{S}, \theta - \varepsilon/2)$ is a FNF ε-approximation of $FSP(\mathcal{D}, \theta)$ with probability at least $1-\delta$.*

Proof. Let us suppose that $\sup_{s\in\mathbb{S}} |f_{\mathcal{D}}(s) - f_{\mathcal{S}}(s)| \leq \varepsilon/2$. In such a scenario, we have that for all sequential patterns $s \in \mathcal{D}$, it results $f_{\mathcal{S}}(s) \in [f_{\mathcal{D}}(s) - \varepsilon/2, f_{\mathcal{D}}(s) + \varepsilon/2]$. This also holds for the sequential patterns in $\mathcal{N} = FSP(\mathcal{S}, \theta - \varepsilon/2)$. Therefore, the set $\mathcal{N}$ satisfies Property 3 from Definition 3. It also means that for all $s \in FSP(\mathcal{D}, \theta)$, $f_{\mathcal{S}}(s) \geq \theta - \varepsilon/2$, then such $s \in \mathcal{N}$ and $\mathcal{N}$ also satisfies Property 1. Now, let $\tilde{s}$ be a sequential pattern such that $f_{\mathcal{D}}(\tilde{s}) < \theta - \varepsilon$. Then, $f_{\mathcal{S}}(\tilde{s}) < \theta - \varepsilon/2$, that is $\tilde{s} \notin \mathcal{N}$, which allows us to conclude that $\mathcal{N}$ also has Property 2 from Definition 3. Since we know that $\sup_{s\in\mathbb{S}} |f_{\mathcal{D}}(s) - f_{\mathcal{S}}(s)| \leq \varepsilon/2$ with probability at least $1-\delta$, then the set $\mathcal{N}$ is a FNF ε-approximation of $FSP(\mathcal{D}, \theta)$ with probability at least $1-\delta$, which concludes the proof. □

Theorem 4 provides a sampling-based algorithm to obtain a FNF ε-approximation of $FSP(\mathcal{D}, \theta)$ with probability $\geq 1-\delta$: take a random sample $\mathcal{S}$ of $|\mathcal{S}| = m$ transactions from $\mathcal{D}$ such that the maximum deviation is bounded by $\varepsilon/2$, that is, $\sup_{s\in\mathbb{S}} |f_{\mathcal{D}}(s) - f_{\mathcal{S}}(s)| \leq \varepsilon/2$; report in output the set $FSP(\mathcal{S}, \theta - \varepsilon/2)$. As illustrated in Section 3.3.1, such a sample size can be computed using an efficient upper bound on the VC-dimension, given in input the desired upper bound on the maximum deviation $\varepsilon/2$ (see Algorithm 2). Algorithm 3 shows the pseudo-code of the sampling algorithm.

We now provide the respective theorem to find a FPF ε-approximation.

Theorem 5. *Given $\varepsilon, \delta \in (0,1)$, let $\mathcal{S}$ be a random sample of size m sequential transactions taken independently at random with replacement from the dataset $\mathcal{D}$ such that $\sup_{s\in\mathbb{S}} |f_{\mathcal{D}}(s) - f_{\mathcal{S}}(s)| \leq \varepsilon/2$ with probability $\geq 1-\delta$. Then, given $\theta \in (0,1]$, the set $FSP(\mathcal{S}, \theta + \varepsilon/2)$ is a FPF ε-approximation of $FSP(\mathcal{D}, \theta)$ with probability $\geq 1-\delta$.*

Proof. Let us suppose that $\sup_{s\in\mathbb{S}} |f_{\mathcal{D}}(s) - f_{\mathcal{S}}(s)| \leq \varepsilon/2$. In such a scenario, we have that for all sequential patterns $s \in \mathcal{D}$, it results $f_{\mathcal{S}}(s) \in [f_{\mathcal{D}}(s) - \varepsilon/2, f_{\mathcal{D}}(s) + \varepsilon/2]$. This also holds for the sequential patterns in $\mathcal{F} = FSP(\mathcal{S}, \theta + \varepsilon/2)$. Therefore, the set $\mathcal{F}$ satisfies Property 3 from Definition 4. It also means that for all $\hat{s} \notin FSP(\mathcal{D}, \theta)$, $f_{\mathcal{S}}(\hat{s}) < \theta + \varepsilon/2$, then such $\hat{s} \notin \mathcal{F}$ and $\mathcal{F}$ also satisfies Property 1. Now, let $\tilde{s}$ be a sequential pattern such that $f_{\mathcal{D}}(\tilde{s}) \geq \theta + \varepsilon$. Then, $f_{\mathcal{S}}(\tilde{s}) \geq \theta + \varepsilon/2$, that is $\tilde{s} \in \mathcal{F}$, which allows us to conclude that $\mathcal{F}$ also has Property 2 from Definition 4. Since we know that $\sup_{s\in\mathbb{S}} |f_{\mathcal{D}}(s) - f_{\mathcal{S}}(s)| \leq \varepsilon/2$ with probability at least $1-\delta$, then the set $\mathcal{F}$ is a FPF ε-approximation of $FSP(\mathcal{D}, \theta)$ with probability at least $1-\delta$, which concludes the proof. □

```
Algorithm 3: SamplingFSP: sampling-based algorithm for FSP mining.
  Data: Dataset $\mathcal{D}$,  $\varepsilon, \delta \in (0,1)$,  $\theta \in (0,1]$.
  Result: Set $\mathcal{N}$ that is a FNF $\varepsilon$-approximation (resp. a FPF $\varepsilon$-approximation) of
          $FSP(\mathcal{D}, \theta)$ with probability $\geq 1 - \delta$.
1 $m \leftarrow$ ComputeSampleSize($\mathcal{D}, \varepsilon, \delta$);
2 $\mathcal{S} \leftarrow$ random sample of $m$ transactions from $\mathcal{D}$;
3 $\mathcal{N} \leftarrow FSP(\mathcal{S}, \theta - \varepsilon/2)$;     /* resp. $\theta + \varepsilon/2$ to obtain a FPF $\varepsilon$-approximation */
4 return $\mathcal{N}$;
```

As explained above, the sample size m can be computed with Algorithm 2 that uses an efficient upper bound on the VC-dimension of sequential patterns, i.e., the s-bound. Then, the sample is generated taking m transactions uniformly and independently at random, with replacement, from $\mathcal{D}$. Finally, the mining of the sample $\mathcal{S}$ can be performed with any efficient algorithm for the exact mining of frequent sequential patterns.

3.5 Experimental Evaluation

In this Section, we report the results of our experimental evaluation on multiple datasets to assess the performance of our sampling algorithm. The goals of the evaluation are the following:

- Assess the performance of our sampling algorithm. In particular, to asses whether with probability $1-\delta$, the sets of frequent sequential patterns extracted from samples are FNF ε-approximations, for the first strategy, and FPF ε-approximations, for the second one, of $FSP(\mathcal{D}, \theta)$. In addition, we compared the performance of the sampling algorithm with the ones to mine the full datasets in term of execution time.

Since no sampling algorithm for rigorously approximating the set of frequent sequential patterns has been previously proposed, we do not consider other methods in our experimental evaluation. In all the experiments, to bound the VC-dimension we used our novel upper bound on the VC-dimension of sequential patterns, i.e., the s-bound (see Definition 7).

3.5.1 Implementation, Environment, and Datasets

The code to compute the bound on the VC-dimension (Algorithm 1), of our sampling algorithm (Algorithm 3), and to perform the evaluation has been developed in Java and executed using version 1.8.0_201. We have performed all our experiments on the same machine with 512 GB of RAM and 2 Intel(R) Xeon(R) CPU E5-2698 v3 @ 2.3GHz, running Ubuntu 14.04. To mine sequential patterns, we used the PrefixSpan [54] implementation provided by the SPMF library [18]. Our open-source implementation and the code developed for the tests, including scripts to reproduce all results, are available at https://github.com/VandinLab/VCRadSPM.

Here, we describe the datasets we used in our evaluation. All the datasets are obtained starting from the following real datasets, publicly available online:[1]

- BIBLE: a conversion of the Bible into sequence where each word is an item;
- BMS1: contains sequences of click-stream data from the e-commerce website Gazelle;
- BMS2: contains sequences of click-stream data from the e-commerce website Gazelle;
- FIFA: contains sequences of click-stream data from the website of the FIFA World Cup 98;
- KOSARAK: contains sequences of click-stream data from an Hungarian news portal;
- LEVIATHAN: is a conversion of the novel Leviathan by Thomas Hobbes (1651) as a sequence dataset where each word is an item;
- MSNBC: contains sequences of click-stream data from MSNBC website and each item represents the category of a web page;
- SIGN: contains sign language utterance.

The typical scenario for the application of sampling is that the dataset to mine is very large, sometimes even too large to fit in the main memory of the machine. Thus, in applying sampling techniques, we aim to reduce the size of such a dataset, considering only a sample of it, in order to obtain an amount of data of reasonable size. Since the number of transactions in each real dataset described above is fairly limited, we replicated each dataset to reach modern datasets sizes. For each real dataset, we fixed a replication factor and we created a new dataset, replicating each transaction in the dataset a number of times equals to the replication factor. We then used such enlarged

[1]https://www.philippe-fournier-viger.com/spmf/index.php?link=datasets.php

Table 3.1 Datasets characteristics. The Table reports: Dataset $\mathcal{D}$: name of the dataset; $|\mathcal{D}|$: number of transactions; Repl. Factor: replication factor; $|\mathcal{I}|$: total number of items; Avg $||\tau||$: average transaction item-length.

Dataset $\mathcal{D}$	$\lvert\mathcal{D}\rvert$	Rep. Factor	$\lvert\mathcal{I}\rvert$	Avg. $\lVert\tau\rVert$
BIBLE	7273800	200x	13905	21.6
BMS1	5960100	100x	497	2.5
BMS2	7751200	100x	3340	4.6
FIFA	4090000	200x	2990	36.2
KOSARAK	6999900	100x	14804	8.0
LEVIATHAN	5835000	1000x	9025	33.8
MSNBC	9898180	10x	17	4.8
SIGN	7300000	10000x	267	52.0

datasets as input for our sampling algorithm. The replication factors we considered are the following: BIBLE and FIFA = 200x; BMS1, BMS2 and KOSARAK = 100x; LEVIATHAN = 1000x; MSNBC = 10x; SIGN = 10000x. The characteristics of the enlarged datasets are reported in Table 3.1.

3.5.2 Sampling Algorithm Results

In this Section, we describe the results obtained with our sampling algorithm (Algorithm 3). As explained above, the typical scenario to apply sampling is that the dataset to mine is very large. Thus, we aim to reduce the size of such a dataset, considering only a sample of it. In addition, from the sample, we aim to obtain a good and rigorous approximation of the results that would have been obtained from the entire dataset, i.e., a FNF or FPF ε-approximation. In all our experiments, we fixed $\varepsilon = 0.01$ and $\delta = 0.1$. The steps of the evaluation are the following (Algorithm 3): given a dataset $\mathcal{D}$ as input, we computed the sample size m, using Algorithm 2, to obtain a FNF $\varepsilon = 0.01$-approximation (resp. FPF 0.01-approximation) with probability at least $1 - \delta = 0.90$. Then, we extracted a random sample $\mathcal{S}$ of m transactions from $\mathcal{D}$ and we ran the algorithm to mine the frequent sequential patterns on $\mathcal{S}$. Finally, we verified whether the set of frequent sequential patterns extracted from the sample was a FNF 0.01-approximation (resp. FPF 0.01-approximation) to $FSP(\mathcal{D}, \theta)$. For each dataset $\mathcal{D}$, we repeated the experiment 5 times, and then we computed the fraction of times the sets of frequent sequential patterns extracted from the samples had the properties described in Definition 3 (resp. Definition 4). Table 3.2 shows the results.

Table 3.2 Sampling algorithms results. The Table reports: Dataset $\mathcal{D}$: name of the dataset; θ: minimum frequency threshold; $|\mathcal{S}|/|\mathcal{D}|$: ratio between the sample size $|\mathcal{S}|$ and the size of the dataset $|\mathcal{D}|$; Max-Err: maximum absolute error over the 5 samples; Avg-Err: average absolute error over the 5 samples; FNF ε-app. (%): percentage of FNF ε-approximations obtained over the 5 samples; FPF ε-app. (%): percentage of FPF ε-approximations obtained over the 5 samples.

Dataset $\mathcal{D}$	θ	$\lvert\mathcal{S}\rvert/\lvert\mathcal{D}\rvert$	Max-Err (x10^{-4})	Avg-Err (x10^{-4})	FNF ε-app. (%)	FPF ε-app. (%)
BIBLE	0.1	0.24	9.33	7.47	100	100
BMS1	0.012	0.17	5.45	4.70	100	100
BMS2	0.012	0.16	4.08	3.14	100	100
FIFA	0.25	0.50	8.68	7.07	100	100
KOSARAK	0.02	0.52	7.18	4.95	100	100
LEVIATHAN	0.15	0.30	9.19	7.84	100	100
MSNBC	0.02	0.37	4.33	3.63	100	100
SIGN	0.4	0.20	14.14	12.19	100	100

From the results, it is possible to notice that the samples obtained from the datasets are about 2 to 5 times smaller than the respective whole datasets. Moreover, in all the runs and for all the datasets, we obtained a FNF ε-approximation (resp. FPF ε-approximation). Such results are even better than the theoretical guarantees, that ensure to obtain such approximations with probability at least 90%. We also reported the maximum absolute error Max-Err $= \max_{i\in\{1,\dots,5\}} \max_{s\in\mathcal{N}_i} |f_{\mathcal{D}}(s) - f_{\mathcal{S}_i}(s)|$ and the average absolute error Avg-Err $= \frac{1}{5}\sum_{i=1}^{5} \max_{s\in\mathcal{N}_i} |f_{\mathcal{D}}(s) - f_{\mathcal{S}_i}(s)|$, where $\mathcal{N}_i$ is the set of frequent sequential patterns extracted from the sample $\mathcal{S}_i$, with $i \in \{1,\dots,5\}$, to obtain a FNF ε-approximation. (Let us note that we considered 5 samples, since we ran each experiment 5 times). They represent, respectively, the maximum and the average, over the 5 runs, of the maximum absolute difference between the frequency that the sequential patterns have in the whole dataset and that they have in the sample, over all the sequential patterns extracted from the sample. Again, the results obtained are better than the theoretical guarantees, that ensure a maximal absolute difference lower than $\varepsilon/2 = 0.005$.

Finally, Figure 3.1 shows the comparison between the average execution time of the sampling algorithm and the average execution time of the mining of the whole dataset, over 5 runs. For all the datasets, the sampling algorithm requires less time than the mining of the whole dataset. For BMS1 and BMS2, the mining of the whole dataset is very fast since the number of frequent sequential patterns extracted from it is low. Thus,

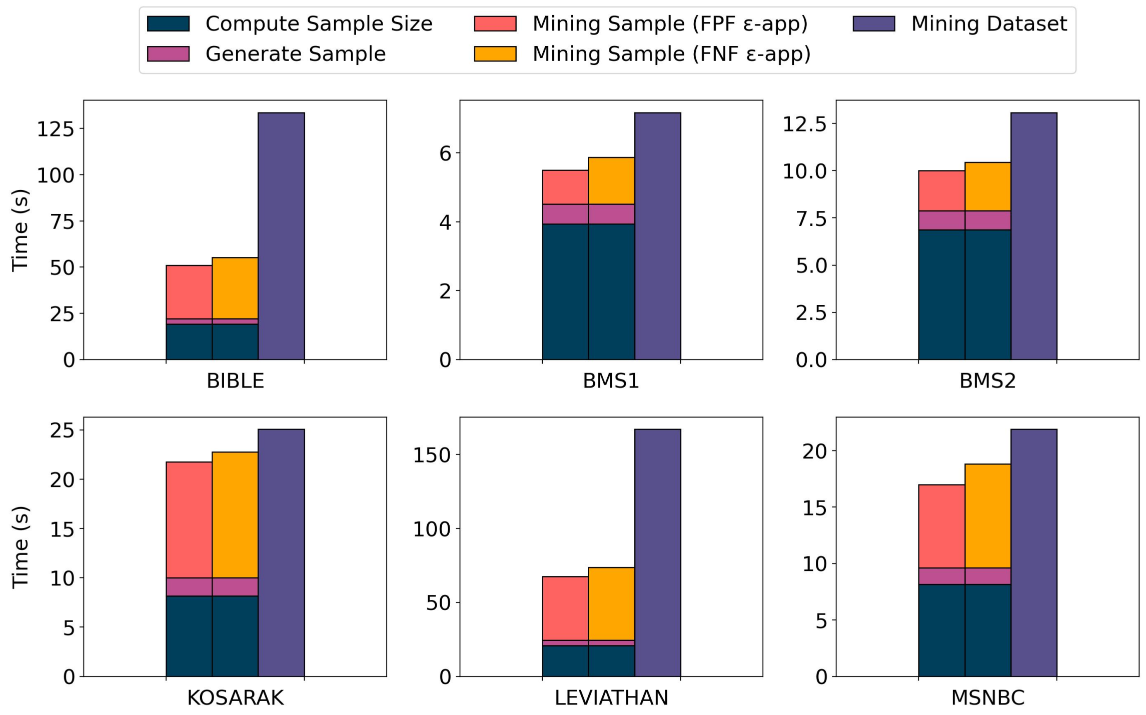

Fig. 3.1 Execution time of the sampling algorithm. It shows: Mining Dataset: the execution time required to mine the whole dataset; the execution times of the sampling algorithm to obtain a FNF or a FPF ε-approximation: Compute Sample Size: the execution time to compute the sample size; Generate Sample: the execution time to generate the sample; Mining Sample: the execution time to mine the sample.

there is not a large difference between the execution time to mine the whole dataset and the execution time for the sampling algorithm, which is most due to the computation of the sample size. Similar results have been obtained with KOSARAK and MSNBC. As expected, for all the datasets, the execution time of the sampling algorithm to obtain a FNF ε-approximation is larger than the one to obtain a FPF ε-approximation, since the minimum frequency threshold used in the first case is lower, resulting in a higher number of extracted sequential patterns.

3.5.3 Analysis of MSNBC

We now briefly discuss some of the sequential patterns extracted from the MSNBC dataset, for which richer information regarding the data is available. In particular, in MSNBC, each transaction contains the sequence of click-stream data generated by a single view on the MSNBC website by a user, and each item represents the category of a visited webpage, e.g., “frontpage”, “news”, “sports”, and so forth.

The two most frequent sequential patterns extracted in the real dataset with a classic FSP algorithm are single categories, that is, sequential patterns of item-length 1: $\langle\{frontpage\}\rangle$ is the most frequent while $\langle\{on - air\}\rangle$ is the second one. They are also the two most frequent sequential patterns extracted in all the five samples using our sampling algorithms. The most frequent sequential patterns with item-length greater than one are the sequential patterns $\langle\{frontpage\}, \{frontpage\}\rangle$ and $\langle\{frontpage\}, \{frontpage\}, \{frontpage\}\rangle$. For $\langle\{frontpage\}, \{frontpage\}\rangle$, in the 75% of the transactions in which it appears, there is at least an instance of such a pattern where the two items are consecutive. This means that users visited two consecutive webpages of the same category, "frontpage", or that they refreshed the same page twice, while in the 25% of the transactions in which it appears, users visited webpages of other categories between the two "frontpage" webpages. Instead, for $\langle\{frontpage\}, \{frontpage\}, \{frontpage\}\rangle$ the percentage of transactions in which the three items are consecutive is 59%. We also observed similar results with other categories: sequential patterns that are sequences of the same item, and so of the same category, have higher frequency. This fact highlights that users usually visit more frequently pages of the same category or that they refresh multiple times the same pages.

The most frequent sequential patterns that are not sequences of the same item are combinations of the items "frontpage" and "news", for example, $\langle\{frontpage\}, \{news\}\rangle$, $\langle\{frontpage\}, \{news\}, \{news\}\rangle$ and $\langle\{news\}, \{frontpage\}\rangle$. Surprisingly, the item "on-air" alone is more frequent that the item "news" alone. This means that users visit "news" webpages coming from a "frontpage" more frequently than "on-air" webpages, though they visit more frequently "on-air" webpages.

Chapter 4

Mining True Frequent Sequential Patterns

In this Chapter, we consider the problem of mining true frequent sequential patterns. While in Chapter 3 we aimed to extract a meaningful random sample from a dataset, in this Chapter we consider the scenario in which the dataset itself is a random sample from an unknown underlying generative process, as happens in many real applications, and by analyzing the dataset, one is interested in mining sequential patterns that are frequently generated by such a process. In this task, unlike the one described in the previous Chapter, the sample, and thus its size, is already fixed. Then, a key challenge is the computation of a bound on the difference between the true frequencies of the sequential patterns w.r.t. the underlying generative distribution and their frequencies in the dataset. Let us note that the quality of such a bound depends on the available amount of data, i.e., the size of the dataset. To solve this difficult task, we use results from the previous Chapter based on the (empirical) VC-dimension of sequential patterns, which allow us to obtain rigorous approximations of the true frequent sequential patterns, with guarantees on the false positives or on the false negatives. The contributions described in this Chapter appear in [73].

4.1 Introduction

In several applications, the analysis of a dataset is performed to gain insight on the *underlying generative process* of the data. For example, in market basket analysis one is interested in gaining knowledge on the behaviour of all the customers, which can be modeled as a generative process from which the transactions in the dataset have been drawn. In such a scenario, one is not interested in sequential patterns that are frequent *in the dataset*, but in sequential patterns that are frequent *in the generative process*, that is, whose probability of appearing in a transaction generated from the process is above a threshold θ. Such patterns, called *true frequent patterns* (see Section 2.3), have been introduced by [68], which provides a VC-dimension based approach to mine true frequent itemsets. While there is a relation between the probability that a pattern appears in a transaction generated from the process and its frequency in the dataset, one cannot simply look at patterns with frequency above θ in the dataset to find the ones with probability above θ in the process. Moreover, due to the stochastic nature of the data, one cannot identify the true frequent patterns with certainty, and approximations are to be sought. In such a scenario, relating the probability that a pattern appears in a transaction generated from the process with its frequency in the dataset using standard techniques is even more challenging than the scenario illustrated in Chapter 3. Hoeffding inequality and union bounds require to bound the number of all the possible sequential patterns that can be generated from the process. Such a bound is infinite if one considers all possible sequential patterns (e.g., does not bound the pattern length). To the best of our knowledge, no method to mine *true frequent sequential patterns* has been proposed.

4.1.1 Our Contributions

In this Chapter, we study the problem of mining *true frequent sequential patterns* and we propose an efficient algorithm for this problem, based on the concept of empirical VC-dimension of sequential patterns. In this regard, our contributions are:

- We formally define the task of mining *true frequent sequential patterns* and two rigorous approximations of the set of true frequent sequential patterns: one with no *false negatives*, that is, containing all elements of the original set; and one with no *false positives*, that is, without any element that is not in the original set.

- We introduce an efficient algorithm to obtain rigorous approximations of the true frequent sequential patterns with probability $1-\delta$, where δ is a confidence parameter set by the user. Our algorithm hinges on a uniform convergence bound and it allows to obtain accurate approximations of the true frequent sequential patterns, where the accuracy depends on the size of the available data.

- We perform an extensive experimental evaluation analyzing several sequential datasets, showing that our algorithm provide high-quality approximations, even better than guaranteed by its theoretical analysis.

4.1.2 Related Work

To the best of our knowledge, [68] is the only work that considers the extraction of frequent patterns w.r.t. an underlying generative process, based on the concept of empirical VC-dimension of itemsets. While we use the general framework introduced by [68], the solution proposed by [68] requires to solve an optimization problem that is tailored to itemsets and, thus, not applicable to sequential patterns. In addition, computing the solution of such a problem could be relatively expensive. A more recent work by Pellegrina and Vandin [59] considers the problem of mining significant patterns under a similar framework, making more realistic assumptions on the underlying generative process compared to commonly used tests (e.g., Fisher's exact test).

Several works have been proposed to identify statistically significant patterns where the significance is defined in terms of the comparison of patterns statistics. Few methods [24, 47, 62] have been proposed to mine statistically significant sequential patterns. These methods are orthogonal to our approach, which focuses on finding sequential patterns that are frequent w.r.t. an underlying generative distribution.

We remind interested readers to the related work of Chapter 3 (Section 3.1.2) for works regarding frequent sequential pattern mining and other pattern mining problems that use the (empirical) VC-dimension.

4.1.3 Organization of the Chapter

The rest of the Chapter is structured as follows. Section 4.2 contains the main definitions and concepts used throughout this Chapter. In Section 4.3, we provide a strategy to compute a probabilistic upper bound on the maximum deviation for the true frequent sequential pattern mining problem based on the empirical VC-dimension of sequential patterns. In Section 4.4, we introduce our algorithm to find a rigorous approximation of the true frequent sequential patterns. Finally, Section 4.5 reports the results of an extensive suite of experiments performed to evaluate the effectiveness and performance of our algorithm to find approximations of the true frequent sequential patterns.

4.2 Preliminaries

We now define the definitions and concepts used throughout the Chapter, refreshing some notions defined in Chapter 2. We start by refreshing the task of sequential pattern mining, introduced in Section 2.1.2, and formally define the problem of mining and approximating the true frequent sequential patterns. We then apply the concept of maximum deviation, introduced in Section 2.4.1, to the true frequent pattern mining scenario.

4.2.1 True Frequent Sequential Pattern Mining

Let us remember, from Section 2.1.2, that a *sequential pattern*, or *sequence*, $s = \langle S_1, S_2, \dots, S_k \rangle$ is a finite ordered sequence of *itemsets* S_i, with $i \in \{1, \dots, k\}$, which are sets of elements called *items*. The *length* $|s|$ of s is the number of itemsets in s, the *item-length* $||s||$ of s is the sum of the sizes of the itemsets in s, that is,

$$||s|| = \sum_{i=1}^{|s|} |S_i|,$$

while the *capacity* $c(s)$ of a sequence s is the number of distinct subsequences of s, i.e., $c(s) = |\{a : a \sqsubseteq s\}|$. Given a *dataset* $\mathcal{D}$, which is a bag of $|\mathcal{D}|$ sequences from a domain $\mathbb{S}$, and a *minimum frequency threshold* $\theta \in (0, 1]$, *frequent sequential pattern* (FSP) mining is the task of reporting the set $FSP(\mathcal{D}, \theta)$ of all the sequences whose frequency in $\mathcal{D}$ is at least θ, and their frequencies, that is,

$$FSP(\mathcal{D}, \theta) = \{(s, f_{\mathcal{D}}(s)) : s \in \mathbb{S}, f_{\mathcal{D}}(s) \geq \theta\},$$

where the *frequency* $f_{\mathcal{D}}(s)$ of s in $\mathcal{D}$ is the *fraction* of transactions in $\mathcal{D}$ to which s belongs, that is,

$$f_{\mathcal{D}}(s) = \frac{|\{\tau \in \mathcal{D} : s \sqsubseteq \tau\}|}{|\mathcal{D}|}.$$

In several real applications, the dataset $\mathcal{D}$ is a sample of transactions independently drawn from an *unknown probability distribution* π on $\mathbb{S}$, that is, $\pi : \mathbb{S} \to [0, 1]$. In such a scenario, the dataset $\mathcal{D}$ is a finite bag of $|\mathcal{D}|$ *independent identically distributed (i.i.d.)* samples from π. For any sequence $s \in \mathbb{S}$, the *true support set* $T(s)$ of s is the set of sequential patterns in $\mathbb{S}$ to which s belongs, i.e., $T(s) = \{\tau \in \mathbb{S} : s \sqsubseteq \tau\}$, and the *true frequency* $t_\pi(s)$ of s w.r.t. π is the probability that a transaction sampled from π contains s, that is,

$$t_\pi(s) = \Pr_{\tau \sim \pi}(s \sqsubseteq \tau).$$

In such a scenario, the final goal of the data mining process on $\mathcal{D}$ is to gain a better understanding of the process that generated the data, i.e., the distribution π, through the true frequencies of the sequential patterns, which are unknown and only approximately reflected in the dataset $\mathcal{D}$. Thus, given a probability distribution π on $\mathbb{S}$ and a minimum frequency threshold $\theta \in (0,1]$, *true frequent sequential pattern (TFSP) mining* is the task of reporting the set $TFSP(\pi,\theta)$ of all sequential patterns whose true frequencies w.r.t. π are at least θ, and their true frequencies, that is,

$$TFSP(\pi,\theta) = \{(s, t_\pi(s)) : s \in \mathbb{S}, t_\pi(s) \geq \theta\}.$$

Let us note that the probability distribution π, and then the true frequencies, are unknown and thus it is not possible to directly mine the set $TFSP(\pi,\theta)$. In addition, given a finite number of random samples from π, i.e., the dataset $\mathcal{D}$, the frequencies of the sequential patterns in the dataset $\mathcal{D}$ only approximately reflect the underlying distribution and thus one has to resort to approximations of $TFSP(\pi,\theta)$. Analogously to the two approximations defined in Chapter 3 for the frequent sequential pattern mining through sampling task, we now define two approximations of $TFSP(\pi,\theta)$, depending on the application one is interested in: the first one that contains all the true frequent sequential patterns, while the second one that does not contain false positives.

Definition 8. *Given $\mu \in (0,1)$, a false negatives free (FNF) μ-approximation $\mathcal{E}$ of $TFSP(\pi,\theta)$ is defined as a set of pairs (s, f_s),*

$$\mathcal{E} = \{(s, f_s) : s \in \mathbb{S}, f_s \in [0,1]\}$$

that has the following properties:

- *$\mathcal{E}$ contains a pair (s, f_s) for every $(s, t_\pi(s)) \in TFSP(\pi,\theta)$;*
- *$\mathcal{E}$ contains no pair (s, f_s) such that $t_\pi(s) < \theta - \mu$;*
- *for every $(s, f_s) \in \mathcal{E}$, it holds $|t_\pi(s) - f_s| \leq \mu/2$.*

Intuitively, the approximation $\mathcal{E}$ contains all the true frequent sequential patterns that are in $TFSP(\pi,\theta)$ (i.e., there are no *false negatives*) and no sequential pattern that has true frequency w.r.t π much below θ. In addition, $\mathcal{E}$ provides a good approximation of the actual true frequency of the sequential patterns, within an error $\mu/2$ which depends on the size of the available data.

Definition 9. *Given $\mu \in (0,1)$, a false positives free (FPF) μ-approximation $\mathcal{M}$ of $TFSP(\pi,\theta)$ is defined as a set of pairs (s, f_s),*

$$\mathcal{M} = \{(s, f_s) : s \in \mathbb{S}, f_s \in [0,1]\}$$

that has the following properties:

- *$\mathcal{M}$ contains no pair (s, f_s) such that $t_\pi(s) < \theta$;*
- *$\mathcal{M}$ contains all the pairs (s, f_s) such that $t_\pi(s) \geq \theta + \mu$;*
- *for every $(s, f_s) \in \mathcal{M}$, it holds $|t_\pi(s) - f_s| \leq \mu/2$.*

The approximation $\mathcal{M}$ does not contain *false positives*, that is, sequences with $t_\pi(s) < \theta$. In addition, it does not miss sequences with $t_\pi(s) \geq \theta + \mu$ and, similarly to the FNF μ-approximation, we have that, for every pair in $\mathcal{M}$, it gives a good approximation of the actual true frequency of the sequential patterns, within an error $\mu/2$ which depends on the size of the available data.

4.2.2 Maximum Deviation

In the true frequent pattern mining task, we aim to find good estimates for $t_\pi(s)$ simultaneously for each sequential pattern $s \in \mathbb{S}$. In such a scenario, the true frequency $t_\pi(s)$ of a sequential pattern s w.r.t. the probability distribution π and its frequency $f_\mathcal{D}(s)$ in $\mathcal{D}$ represent, respectively, the expectation and the empirical average of a function associated with the sequential pattern s, since

$$t_\pi(s) = \mathbb{E}_{\tau \sim \pi}[\mathbb{1}_\tau(s)]$$

and

$$f_\mathcal{D}(s) = \frac{1}{|\mathcal{D}|} \sum_{\tau \in \mathcal{D}} \mathbb{1}_\tau(s),$$

where $\mathbb{1}_\tau(s)$ the indicator function that assumes the value 1 if and only if $s \sqsubseteq \tau$. Thus, in such a scenario the maximum deviation (see Equation 2.1) is

$$\sup_{s \in \mathbb{S}} |t_\pi(s) - f_\mathcal{D}(s)|.$$

In the next Section, we provide a strategy, based on the concept of empirical VC-dimension of sequential patterns, to compute a probabilistic upper bound on the maximum deviation in the true frequent sequential pattern mining scenario.

4.3 Bound on the Max Deviation for TFSP Mining

In this Section, we provide a strategy to compute an upper bound on the maximum deviation $\mu/2$ for the true frequent sequential pattern mining problem, that is, $\sup_{s \in \mathbb{S}} |t_\pi(s) - f_\mathcal{D}(s)| \leq \mu/2$, using an upper bound on the empirical VC-dimension (see

Section 2.4.2). Such a result underlies the strategy for mining the true frequent sequential patterns that will be introduced in Section 4.4.

Let us remember that a range space is a pair $(X, \mathcal{R})$ where X contains points and $\mathcal{R}$ contains ranges. For a domain $\mathbb{S}$, X is the domain itself, while $\mathcal{R}$ contains the sequential transactions that are the true support set for some sequential patterns.

Definition 10. *Let $\mathbb{S}$ be a domain. We define $RS = (X, \mathcal{R})$ to be a range space associated with $\mathbb{S}$ such that:*

- *$X = \mathbb{S}$ is the set of sequences in $\mathbb{S}$;*
- *$\mathcal{R} = \{T(s) : s \in \mathbb{S}\}$ is a family of sets of sequences such that for each sequential pattern s, the set $T(s) = \{\tau \in \mathbb{S} : s \sqsubseteq \tau\}$ is the true support set of s.*

The s-index (see Definition 6) defined in [75] and the s-bound (see Definition 7) defined in Chapter 3 of this Thesis, are upper bounds on the empirical VC-dimension $EVC(RS, \mathcal{D})$ of the range space associated with $\mathbb{S}$ computed on $\mathcal{D}$.

Theorem 6. *Let $RS = (X, \mathcal{R})$ be the range space associated with $\mathbb{S}$, let π be a probability distribution on the domain $\mathbb{S}$, and let $\mathcal{D}$ be a finite bag of $|\mathcal{D}|$ i.i.d. samples from π. If the empirical VC-dimension $EVC(RS, \mathcal{D})$ of RS on $\mathcal{D} \leq d$, then, given $\delta \in (0, 1)$ and*

$$\mu = \sqrt{\frac{2}{|\mathcal{D}|}\left(d + \ln\frac{1}{\delta}\right)},$$

$\sup_{s\in\mathbb{S}} |t_\pi(s) - f_\mathcal{D}(s)| \leq \mu/2$ *with probability at least* $1 - \delta$.

Proof. From Theorem 1, we know that $\mathcal{D}$ is a $\mu/2$-bag of $\mathbb{S}$ w.r.t. π with probability at least $1 - \delta$. Then, from Definition 2,

$$\left|\Pr_\pi(r) - \frac{|\mathcal{D} \cap r|}{|\mathcal{D}|}\right| \leq \frac{\mu}{2}$$

for all $r \in \mathcal{R}$. Given a sequential pattern $s \in \mathbb{S}$ and its real support set $T(s)$, which is the range r_s, from the definition of range space associated with $\mathbb{S}$, we have

$$\Pr_\pi(r_s) = t_\pi(s)$$

and

$$\frac{|\mathcal{D} \cap r_s|}{|\mathcal{D}|} = f_\mathcal{D}(s).$$

Thus, $\sup_{s\in\mathbb{S}} |t_\pi(s) - f_\mathcal{D}(s)| \leq \mu/2$ with probability at least $1 - \delta$, which concludes the proof. □

```
Algorithm 4: ComputeMaxDev: compute an upper bound on the max deviation.
    Data: Dataset $\mathcal{D}$, $\delta \in (0,1)$.
    Result: Upper bound on the max deviation $\mu/2$ s.t.
            $\Pr\left(\sup_{s\in\mathbb{S}} |t_\pi(s) - f_{\mathcal{D}}(s)| \leq \mu/2\right) \geq 1-\delta$.
  1 $d \leftarrow$ SBoundUpp($\mathcal{D}$);
  2 $\mu \leftarrow \sqrt{2/|\mathcal{D}|\,(d + \ln(1/\delta))}$;
  3 return $\mu/2$;
```

Algorithm 4 using an efficient upper bound on the VC-dimension of sequential patterns, i.e., the s-bound (see Section 3.3, Algorithm 1), shows how to compute an upper bound on the maximum deviation which guarantees that $\sup_{s\in\mathbb{S}} |t_\pi(s) - f_{\mathcal{D}}(s)| \leq \mu/2$ with probability $\geq 1-\delta$. Let us note that while here we employ the s-bound to upper bound the empirical VC-dimension computed on $\mathcal{D}$, every upper bound on such a measure can be employed. This algorithm is used in our strategy to mine rigorous approximations of the true frequent sequential patterns (Algorithm 5 of Section 4.4).

4.4 Algorithm for TFSP Mining

In this Section, we describe our approach to find rigorous approximations of the TFSPs. In particular, given a dataset $\mathcal{D}$, that is a finite bag of $|\mathcal{D}|$ i.i.d. samples from an unknown probability distribution π on $\mathbb{S}$, a minimum frequency threshold $\theta \in (0,1]$ and a confidence parameter $\delta \in (0,1)$, we aim to find rigorous approximations of $TFSP(\pi,\theta)$, i.e., FNF μ-approximations (Definition 8) and FPF μ-approximations (Definition 9), with probability at least $1-\delta$.

The intuition behind our approach is the following. If we know an upper bound $\mu/2$ on the maximum deviation, i.e., $\sup_{s\in\mathbb{S}} |t_\pi(s) - f_{\mathcal{D}}(s)| \leq \mu/2$, we can identify a frequency threshold $\tilde{\theta}$ (resp. $\hat{\theta}$) such that the set $FSP(\mathcal{D},\tilde{\theta})$ is a FNF μ-approximation (resp. $FSP(\mathcal{D},\hat{\theta})$ is a FPF μ-approximation) of $TFSP(\pi,\theta)$. The upper bound on the maximum deviation can be computed, as illustrated in the previous Sections, with the empirical VC-dimension.

We now describe how to identify the threshold $\tilde{\theta}$ that allows to obtain a FNF μ-approximation. Suppose that $\sup_{s\in\mathbb{S}} |t_\pi(s) - f_{\mathcal{D}}(s)| \leq \mu/2$. In such a scenario, we have that every sequential pattern $s \in TFSP(\pi,\theta)$, and so that has $t_\pi(s) \geq \theta$, has a frequency $f_{\mathcal{D}}(s) \geq \theta - \mu/2 = \tilde{\theta}$. Hence, by mining the sequential patterns that have frequency in $\mathcal{D}$ greater or equal to $\tilde{\theta} = \theta - \mu/2$, we do not miss any TFSPs, avoiding false negatives. The following theorem formalizes the strategy to obtain a FNF μ-approximation.

Theorem 7. *Given* $\delta \in (0, 1)$*, such that* $\sup_{s\in\mathbb{S}} |t_\pi(s) - f_{\mathcal{D}}(s)| \leq \mu/2$ *with probability at least* $1 - \delta$*, and given* $\theta \in (0, 1]$*, the set* $FSP(\mathcal{D}, \tilde{\theta})$*, with* $\tilde{\theta} = \theta - \mu/2$*, is a FNF* μ*-approximation of the set* $TFSP(\pi, \theta)$ *with probability at least* $1 - \delta$*.*

Proof. Let us suppose that $\sup_{s\in\mathbb{S}} |t_\pi(s) - f_{\mathcal{D}}(s)| \leq \mu/2$. Thus, we have that for all the sequential patterns $s \in \mathbb{S}$, it results $f_{\mathcal{D}}(s) \in [t_\pi(s) - \mu/2, t_\pi(s) + \mu/2]$. This also holds for the sequential patterns in $\mathcal{E} = FSP(\mathcal{D}, \tilde{\theta})$. Therefore, the set $\mathcal{E}$ satisfies Property 3 of Definition 8. It also means that for all $s \in TFSP(\pi, \theta)$, $f_{\mathcal{D}}(s) \geq \theta - \mu/2 = \tilde{\theta}$, that is, $s \in \mathcal{E}$, which allows us to conclude that $\mathcal{E}$ also has Property 1 from Definition 8. Now, let s^* be a sequential pattern such that $t_\pi(s^*) < \theta - \mu$. Then, $f_{\mathcal{D}}(s^*) < \theta - \mu/2$, that is $s^* \notin \mathcal{E}$, which allows us to conclude that $\mathcal{E}$ also has Property 2 from Definition 8. Since we know that $\sup_{s\in\mathbb{S}} |t_\pi(s) - f_{\mathcal{D}}(s)| \leq \mu/2$ with probability at least $1 - \delta$, then the set $\mathcal{E}$ is a FNF μ-approximation of $TFSP(\pi, \theta)$ with probability at least $1-\delta$, which concludes the proof. □

Theorem 7 provides a strategy to obtain a FNF μ-approximation of $TFSP(\pi, \theta)$ with probability $\geq 1 - \delta$: compute an upper bound on the maximum deviation such that $\sup_{s\in\mathbb{S}} |t_\pi(s) - f_{\mathcal{D}}(s)| \leq \mu/2$ with probability $\geq 1 - \delta$; report in output the set $FSP(\mathcal{D}, \theta - \mu/2)$. As illustrated in Section 4.3, such a probabilistic upper bound on the maximum deviation can be computed using an upper bound on the empirical VC-dimension. In particular, in Algorithm 5 we show the pseudo-code to mine a FNF μ-approximation of true frequent sequential patterns using the s-bound (see Section 3.3, Algorithm 1).

We now provide the respective theorem to find a FPF μ-approximation.

Theorem 8. *Given* $\delta \in (0, 1)$*, such that* $\sup_{s\in\mathbb{S}} |t_\pi(s) - f_{\mathcal{D}}(s)| \leq \mu/2$ *with probability at least* $1 - \delta$*, and given* $\theta \in (0, 1]$*, the set* $FSP(\mathcal{D}, \hat{\theta})$*, with* $\hat{\theta} = \theta + \mu/2$*, is a FPF* μ*-approximation of the set* $TFSP(\pi, \theta)$ *with probability at least* $1 - \delta$*.*

Proof. Suppose that $\sup_{s\in\mathbb{S}} |t_\pi(s)-f_{\mathcal{D}}(s)| \leq \mu/2$. Thus, we have that for all the sequential patterns $s \in \mathbb{S}$, it results $f_{\mathcal{D}}(s) \in [t_\pi(s) - \mu/2, t_\pi(s) + \mu/2]$. This also holds for the sequential patterns in $\mathcal{M} = FSP(\mathcal{D}, \hat{\theta})$. Therefore, the set $\mathcal{M}$ satisfies Property 3 of Definition 9. Let s^* be a sequential pattern such that $t_\pi(s^*) < \theta$, that is, it is not a true frequent sequential pattern w.r.t. θ. Then, $f_{\mathcal{D}}(s^*) < \theta + \mu/2 = \hat{\theta}$, that is, $s^* \notin \mathcal{G}$, which allows us to conclude that $\mathcal{M}$ also has Property 1 from Definition 9. Now, let s' be a sequential pattern such that $t_\pi(s') \geq \theta + \mu$. Then, $f_{\mathcal{D}}(s') \geq \theta + \mu/2$, that is $s' \in \mathcal{M}$, which allows us to conclude that $\mathcal{M}$ also has Property 2 from Definition 9. Since we know that $\sup_{s\in\mathbb{S}} |t_\pi(s) - f_{\mathcal{D}}(s)| \leq \mu/2$ with probability at least $1 - \delta$, then the set $\mathcal{M}$ is a FPF μ-approximation of $TFSP(\pi, \theta)$ with probability at least $1 - \delta$, which concludes the proof. □

Algorithm 5: `ApproxTSFP`: Find a μ-approximation of the TFSPs.

```
Data: Dataset $\mathcal{D}$, $\delta \in (0, 1)$, $\theta \in (0, 1]$.
Result: Set $\mathcal{E}$ that is a FNF $\mu$-approximation (resp. FPF $\mu$-approximation) of
        $TFSP(\pi, \theta)$ with probability $\geq 1 - \delta$.
1 $\mu/2 \leftarrow$ ComputeMaxDev($\mathcal{D}, \delta$);
2 $\mathcal{E} \leftarrow FSP(\mathcal{D}, \theta - \mu/2)$;    /* resp. $\theta + \mu/2$ to obtain a FPF $\mu$-approximation */
3 return $\mathcal{E}$;
```

Algorithm 5 shows the pseudo-code of our strategy to find a μ-approximation of the true frequent sequential patterns. Let us note that to compute an upper bound on the maximum deviation, it employs Algorithm 4 based on the empirical VC-dimension of sequential patterns. However, all the reasoning are still valid when other techniques are used to compute an upper bound on the maximum deviation, e.g., the Rademacher complexity [12]. Finally, the mining of $\mathcal{D}$ can be performed with any efficient algorithm for the exact mining of frequent sequential patterns.

4.5 Experimental Evaluation

In this Section, we report the results of our experimental evaluation on multiple datasets to assess the performance of our algorithm to mine true frequent sequential patterns. The goals of the evaluation are the following:

- Assess whether the sequential patterns mined from the datasets with a classical algorithm to mine frequent sequential patterns contain false positives or false negatives w.r.t. the set of true frequent sequential patterns;

- Assess the performance of our algorithm for mining the true frequent sequential patterns. In particular, to assess whether with probability $1 - \delta$ the set of frequent sequential patterns extracted from the dataset with the corrected threshold is a FNF μ-approximation of $TSFP(\pi, \theta)$, for the first method, or it is a FPF μ-approximation of $TSFP(\pi, \theta)$, for the second method.

Since no algorithm to mine true frequent sequential patterns has been previously proposed, we do not consider other methods in our experimental evaluation. In all the experiments, to bound the maximum deviation we used the empirical VC-dimension. In particular, we used our novel upper bound on the empirical VC-dimension of sequential patterns, i.e., the s-bound (see Section 3.3, Definition 7).

4.5.1 Implementation, Environment, and Datasets

The code of our algorithm (Algorithm 5) to mine true frequent sequential patterns, and to perform the evaluation has been developed in Java and executed using version 1.8.0_201. We have performed all our experiments on the same machine with 512 GB of RAM and 2 Intel(R) Xeon(R) CPU E5-2698 v3 @ 2.3GHz, running Ubuntu 14.04. To mine sequential patterns, we used the PrefixSpan [54] implementation provided by the SPMF library [18]. Our open-source implementation and the code developed for the tests, including scripts to reproduce all results, are available at https://github.com/VandinLab/VCRadSPM.

Here, we describe the datasets we used in our evaluation. All the datasets are obtained starting from the following real datasets, publicly available online:[1]

- BIBLE: a conversion of the Bible into sequence where each word is an item;
- BMS1: contains sequences of click-stream data from the e-commerce website Gazelle;
- BMS2: contains sequences of click-stream data from the e-commerce website Gazelle;
- KOSARAK: contains sequences of click-stream data from an Hungarian news portal;
- LEVIATHAN: is a conversion of the novel Leviathan by Thomas Hobbes (1651) as a sequence dataset where each word is an item;
- MSNBC: contains sequences of click-stream data from MSNBC website and each item represents the category of a web page.

To evaluate our algorithm to mine the true frequent sequential patterns, we need to know which are the sequential patterns that are frequently generated from the unknown generative process π. In particular, we need a *ground truth* of the true frequencies of the sequential patterns. We generated pseudo-artificial datasets by taking the original datasets shown in Table 4.1 as ground truth for the true frequencies of the sequential patterns. For each ground truth, i.e., original dataset, we created 4 new datasets by sampling sequential transactions uniformly at random from the original dataset. All the

[1]https://www.philippe-fournier-viger.com/spmf/index.php?link=datasets.php

Table 4.1 Datasets characteristics. The Table reports: Dataset $\mathcal{D}$: name of the dataset; $|\mathcal{D}|$: number of transactions; $|\mathcal{I}|$: total number of items; Avg $||\tau||$: average transaction item-length.

Dataset $\mathcal{D}$	$\|\mathcal{D}\|$	$\|\mathcal{I}\|$	Avg. $\|\|\tau\|\|$
BIBLE	36369	13905	21.6
BMS1	59601	497	2.5
BMS2	77512	3340	4.6
KOSARAK	69999	14804	8.0
LEVIATHAN	5835	9025	33.8
MSNBC	989818	17	4.8

new datasets have the same number of transactions of the respective original datasets. We then used the original datasets as ground truth and we executed our algorithm in the new (sampled) datasets. Therefore, the true frequency of a sequential pattern is its frequency in the original dataset, that is, its frequency in the original dataset is exactly the same that such a pattern would have in an hypothetical infinite number of transactions generated by the unknown generative process π.

4.5.2 True Frequent Sequential Patterns Results

In this Section, we describe the results of our algorithms for mining the true frequent sequential patterns. In all these experiments, we fixed $\delta = 0.1$.

First of all, for each real dataset, we generated 4 pseudo-artificial datasets $\mathcal{D}_i$, $i \in \{1, \dots, 4\}$ from the same ground truth. We then mined the set $FSP(\mathcal{D}_i, \theta)$ and compared it with the TFSPs, that is, the set $FSP(\mathcal{D}, \theta)$, where $\mathcal{D}$ is the corresponding ground truth. Such experiments aimed to verify whether the sets of the FSPs extracted from the pseudo-artificial datasets contain false positives and miss some TFSPs. Table 4.2 shows the fractions of times that the set $FSP(\mathcal{D}_i, \theta)$ contains at least one false positive and misses at least one TFSP from the ground truth. We ran this evaluation over the 4 datasets $\mathcal{D}_i$, $i \in \{1, \dots, 4\}$, of the same size of the corresponding ground truth and we reported the average. For each dataset, we report the results with two frequency thresholds θ. In almost all the cases, the FSPs mined from the pseudo-artificial datasets contain false positives and miss some TFSPs. In particular, with lower frequency thresholds (and, therefore, a larger number of sequential patterns), the fraction of times we found false positives and false negatives usually increases. These results emphasize that, in general, mining FSPs is not enough to learn interesting features of the underlying generative process of the data, and techniques like the one introduced in this Chapter are necessary.

Table 4.2 Average fraction of times that $FSP(\mathcal{D}_i, \theta)$ contains false positives and false negatives. The Table reports: Dataset $\mathcal{D}$; name of the dataset; θ: minimum frequency threshold; |TFSP|: number of true frequent sequential patterns in the ground truth; Times FNs (%): percentage of runs that contain at least one false negative (FN); Times FPs: percentage of runs that contain at least one false positive (FP).

Dataset $\mathcal{D}$	θ	|TFSP|	Times FNs (%)	Times FPs (%)
BIBLE	0.1	174	100	50
	0.05	774	100	100
BMS1	0.025	13	0	50
	0.0225	17	25	0
BMS2	0.025	10	0	0
	0.0225	11	0	0
KOSARAK	0.06	23	0	100
	0.04	41	25	50
LEVIATHAN	0.15	225	100	75
	0.1	651	100	100
MSNBC	0.02	97	25	75
	0.015	143	50	100

Finally, we evaluated the performance of our strategy to mine μ-approximations of the true frequent sequential patterns, with guarantees on the false negatives or on the false positives. From each pseudo-artificial dataset $\mathcal{D}_i$, with $i \in \{1, \ldots, 4\}$, we mined the FSPs using $\tilde{\theta} = \theta - \mu/2$, for the first strategy, i.e., $FSP(\mathcal{D}_i, \tilde{\theta})$, and $\hat{\theta} = \theta + \mu/2$, for the second one, i.e., $FSP(\mathcal{D}_i, \hat{\theta})$, computed, respectively, using Theorem 7 and 8, and we compared the extracted sequential patterns with the TFSPs from the respective ground truth $\mathcal{D}$, i.e., $FSP(\mathcal{D}, \theta)$. Table 4.3 shows the obtained results. Our algorithm performs better than the theoretical guarantees for both type of approximations, indeed we obtained a μ-approximation in all the runs, while the theory guarantees a probability of at least $1 - \delta = 0.9$. We also computed the average fraction of TFSPs reported in the output w.r.t. the total number of sequential patterns reported by the algorithm, that is, $|TFSP|/|FSP(\mathcal{D}_i, \tilde{\theta})|$ for the FNF μ-approximations and $|FSP(\mathcal{D}_i, \hat{\theta})|/|TFSP|$ for the FPF μ-approximations. In particular, for the FNF μ-approximations, such a measure shows the ratio of reported sequential patterns that are not false positives, while for the FPF μ-approximations, it shows the ratio of reported sequential patterns that are true frequent sequential patterns. Let us note that we aimed to obtain ratios as close to 1 as possible but that our strategy does not provide theoretical guarantees in such

Table 4.3 Results of our algorithm to mine μ-approximations of the TFSPs. The Table reports: Dataset $\mathcal{D}$; name of the dataset; θ: minimum frequency threshold; |TFSP|: number of true frequent sequential patterns in the ground truth; FNF μ-app. (%): percentage of FNF μ-approximations obtained over the 4 datasets; $|TFSP|/|FSP(\mathcal{D}_i, \tilde{\theta})|$: average ratio of reported TFSPs over the 4 datasets; FPF μ-app. (%): percentage of FPF μ-approximations obtained over the 4 samples; $|FSP(\mathcal{D}_i, \hat{\theta})|/|TFSP|$: average ratio of reported TFSPs over the 4 datasets.

Dataset $\mathcal{D}$	θ	|TFSP|	FNF μ-app. (%)	|TFSP|/ |FSP($\mathcal{D}_i, \tilde{\theta}$)|	FPF μ-app. (%)	|FSP($\mathcal{D}_i, \hat{\theta}$)|/ |TFSP|
BIBLE	0.1	174	100	0.63	100	0.55
	0.05	774	100	0.33	100	0.32
BMS1	0.025	13	100	0.21	100	0.38
	0.0025	17	100	0.19	100	0.29
BMS2	0.025	10	100	0.32	100	0.13
	0.0025	11	100	0.19	100	0.18
KOSARAK	0.06	23	100	0.64	100	0.41
	0.04	41	100	0.49	100	0.43
LEVIATHAN	0.15	225	100	0.30	100	0.30
	0.1	651	100	0.11	100	0.18
MSNBC	0.02	97	100	0.77	100	0.56
	0.015	143	100	0.65	100	0.50

a direction. For some datasets, e.g., BMS1, BMS2, and LEVIATHAN, such ratios are not very high. Let us note that LEVIATHAN is a very small dataset while BMS1 and BMS2, even if are larger than other datasets, contain very short transactions. Instead with other (larger) datasets, e.g., BIBLE, KOSARAK, and MSNBC, the ratios are higher. Indeed, as previously stated, our strategy crucially hinges on a bound on the maximum deviation, whose quality depends on the available amount of data. (Let us note that except MSNBC, all the other datasets have a fairly limited number of transactions.)

Overall, the results show that our algorithm is a valid strategy to obtain rigorous approximations, with no false negatives or no false positives, for the true frequent sequential pattern mining task, and approaches like the one introduced in this Chapter are necessary to obtain rigorous guarantees.

Chapter 5

Mining Statistically Robust Patterns from a Sequence of Datasets

In this Chapter, we consider the problem of mining statistically robust patterns from a sequence of datasets. Let us remember that in Chapter 4, we considered the scenario in which the dataset is a random sample from an unknown underlying generative process and by analyzing the dataset, one is interested in mining sequential patterns that are frequently generated by such a process. In this Chapter, we consider a similar scenario with a sequence of datasets, which are random samples from (possibly different) unknown underlying generative processes, instead of a single dataset. In such a scenario, we are interested in mining patterns whose true frequencies follow well specified trends through the sequence of datasets and we describe this task for a generic pattern mining task. Again, a key challenge is the computation of a bound on the difference between the true frequencies of the patterns w.r.t. the underlying generative distributions and their frequencies in the datasets. To solve this difficult task, we generalize results from the previous Chapter based on the VC-dimension, which allow us to obtain rigorous approximations of the statistically robust patterns, with guarantees on the false positives or on the false negatives. However, let us note that our strategy can employ any uniform convergence bound. Part of the contributions described in this Chapter appear in [84], while an extended version is currently under review in the journal Knowledge and Information Systems, invited among the best papers accepted at IEEE ICDM'20.

Grosso: (Italian adj.) large, big, robust.

5.1 Introduction

Let us remember, from Section 2.1, that frequent pattern mining [27] is one of the fundamental tasks in data mining, and requires to identify all patterns appearing in fractions at least θ of all transactions from a transactional dataset. Several variants of the problem have explored different types of patterns (from itemsets [2] to sequential patterns [3], to subgroups [36], to graphlets [4]) relevant to applications ranging from market basket analysis to recommendation systems to spam detection.

In several real applications, a pattern is studied in the context of a *sequence of datasets*, where the sequence is given, for example, from the collection of the data at different time points. For example, in market basket analysis, it is natural to study the patterns (e.g., itemsets) in datasets obtained from transactions in different weeks or months. In almost all applications, one can assume that each dataset is obtained from a *generative process* on transactions, which generates transactions according to some probability distribution, as assumed by *statistically-sound pattern mining* [26]. Let us consider, for example, a series of n surveys performed in n different time intervals in a supermarket, where we collect the data of the receipts of the costumers. The goal of such surveys is to infer information on how the behavior of the entire customers population evolves, but, obviously, it is impossible to collect the receipts of the whole population. Thus, our datasets only represent a collection of samples from the whole population.

In such a scenario, patterns of interest are the ones whose probability of appearing in a transaction follows some well-specified trend (e.g., it increases, decreases, or is constant across datasets). In the survey example above, we may be interested in finding sequences of purchases (i.e., sequential patterns) which become more and more common in time to understand how the customers' behavior changes over time. However, the identification of such patterns is extremely challenging, since the underlying probability distributions are unknown and the observed frequencies of the patterns in the data only approximately reflect such probabilities. As a result, considering the same trends at the level of observed frequencies leads to reporting several false positives. This problem is exacerbated by the huge number of potential candidates, which poses a severe *multiple hypothesis correction problem* [58]. In addition, techniques developed for significant pattern mining [26] or for statistically emerging pattern mining [38] can only be applied to (a sequence of) two datasets.

To address such challenges, in this Chapter we introduce a novel framework to identify *statistically robust patterns* from a sequence of datasets, i.e., patterns whose probability of appearing in transactions follows a well-specified trend, while providing guarantees on the quality of the reported patterns in terms of false positives or in terms of false negatives.

5.1.1 Our Contributions

In this Chapter, we introduce the problem of mining *statistically robust patterns* from a *sequence of datasets*. In this regard, our contributions are:

- We define the problem of mining statistically robust patterns, and define an approximation of such patterns that does not contain *false positives*. We also describe three general types of patterns (emerging, descending, and stable) which are of interest in most scenarios.
- We introduce an algorithm, gRosSo, to obtain a rigorous approximation, without false positives, of the statistically robust patterns from a sequence of datasets with probability at least $1 - \delta$, where δ is a confidence parameter set by the user. Our strategy is based on the concept of maximum deviation and can employ any uniform convergence bound (see Section 2.4). We show how such a strategy can be used to approximate the three types of statistically robust patterns we introduced.
- We define an approximation of the statistically robust patterns that does not contain *false negatives*, and explain how gRosSo can be modified to obtain such an approximation with high probability. We also discuss and prove additional guarantees that can be obtained with gRosSo.
- We apply the general framework of statistically robust patterns to mine sequential patterns. We also introduce a novel algorithm to compute an upper bound on the capacity of a sequence that can be used to bound the maximum deviation using the statistical learning concept of VC-dimension of sequential patterns.
- We apply the general framework of statistically robust patterns to mine itemsets using the VC-dimension of itemsets to bound the maximum deviation.
- We perform an extensive experimental evaluation, mining statistically robust sequential patterns and itemsets from pseudo-artificial datasets. Our evaluation proves that relying on frequency alone leads to several spurious discoveries, while

gRosSo provides high-quality approximations for both data mining tasks. Finally, we analyze real datasets mining statistically robust sequential patterns, proving that gRosSo is able to detect various type of patterns.

5.1.2 Related Work

In significant pattern mining (see Section 2.5) the dataset is seen as a sample from an unknown distribution and one is interested in finding patterns significantly deviating from an assumed null distribution (or hypothesis). Many variants and algorithms have been proposed for the problem. We point interested readers to the survey [26] and the recent works [45, 59, 60]. Few works have been proposed to mine statistically significant sequential patterns [24, 47, 83]. These methods are orthogonal to our approach, which focuses on finding patterns whose frequencies w.r.t. underlying generative distributions respect well defined conditions through a sequence of datasets.

The first work that proposed the problem of mining emerging patterns is [14]. To the best of our knowledge, the only work that considers the problem of finding emerging patterns considering a data generative process and provides statistical guarantees is [38]. However, the proposed approach only works with two datasets and only finds patterns with significant differences in the two datasets. Instead, our approach describes more general trends of the probabilities of the patterns and considers more than two datasets, and it is unclear whether the approach of [38] can be modified to work in our scenario.

Few works have been proposed to mine robust patterns, where the robustness is usually defined by constraints between the relation of the observed frequency of a pattern in a dataset and the frequencies of its sub- or super-patterns. For example, [97] defines robust patterns as patterns for which, by removing some of their sub-patterns, the ratio between its original frequency and the frequency of the resulting pattern in a dataset is greater than a user defined parameter. Egho et al. [15] introduces a space of rules patterns model and it defines a Bayesian criterion for evaluating the interest of sequential patterns for mining sequential rule patterns for classification purpose. Differently from our work, these contributions focus on a single dataset and do not consider a dataset as a collection of samples from an unknown generative process.

Finally, let us note that while in this Chapter we use a general framework similar to the one we propose in Chapter 4 to mine true frequent sequential patterns, in this Chapter we consider the problem of mining statistically robust patterns in a sequence of datasets, that is a different task.

We remind interested readers to the related work of Chapter 3 (Section 3.1.2) and Chapter 4 (Section 4.1.2) for works regarding frequent sequential pattern mining, true frequent pattern mining, and other pattern mining problems based on the (empirical) VC-dimension.

5.1.3 Organization of the Chapter

The rest of the Chapter is structured as follows. Section 5.2 contains the definitions and concepts used throughout this Chapter. Our framework for statistically robust pattern mining is presented in Section 5.3. Section 5.4 describes our algorithm, gRosSo to mine statistically robust patterns and provides discussions and proofs of the guarantees that can be obtained with gRosSo. The application of our approach for mining statistically robust *sequential* patterns is described in Section 5.5, while the application for mining statistically robust *itemsets* is described in Section 5.6. Section 5.7 reports the results of an extensive suite of experiments performed to evaluate the effectiveness of gRosSo on pseudo-artificial and real datasets.

5.2 Preliminaries

We now provide the definitions and the concepts used throughout the Chapter, refreshing some notions defined in Chapter 2. We start by refreshing the task of pattern mining and the assumptions of the true frequent pattern mining task, introduced, respectively, in Section 2.1 and 2.3. We then show how to bound the maximum deviation for the task we consider in this Chapter, generalizing the concepts introduced in Chapter 4 for sequential pattern mining to a general pattern mining task.

5.2.1 Pattern Mining

Let us remember, from Section 2.1, that a *pattern* p is an element from a domain $\mathbb{U}$, potentially with some constraints. Given a *dataset* $\mathcal{D}$, which is a bag of $|\mathcal{D}|$ transactions (i.e., patterns), and a *minimum frequency threshold* $\theta \in (0,1]$, *frequent pattern (FP) mining* is the task of reporting the set $FP(\mathcal{D},\theta)$ of all patterns whose frequency $f_{\mathcal{D}}(p)$ in $\mathcal{D}$ is at least θ, and their frequencies, that is,

$$FP(\mathcal{D},\theta) = \{(p, f_{\mathcal{D}}(p)) : p \in \mathbb{U}, f_{\mathcal{D}}(p) \geq \theta\}.$$

where the *frequency* $f_{\mathcal{D}}(p)$ of p in $\mathcal{D}$ is the fraction of transactions in $\mathcal{D}$ to which p belongs, that is,

$$f_{\mathcal{D}}(p) = \frac{|\{\tau \in \mathcal{D} : p \sqsubseteq \tau\}|}{|\mathcal{D}|}.$$

As assumed in true frequent pattern mining (see Section 2.3), in several applications, the dataset $\mathcal{D}$ is a sample of transactions independently drawn from an unknown probability distribution π on $\mathbb{U}$, that is, $\pi : \mathbb{U} \to [0,1]$. In such a scenario, the dataset $\mathcal{D}$ is a finite bag of $|\mathcal{D}|$ i.i.d. samples from π. For any pattern $p \in \mathbb{U}$, the *true support set* $T(p)$ of p is the set of patterns in $\mathbb{U}$ to which p belongs, i.e., $T(p) = \{\tau \in \mathbb{U} : p \sqsubseteq \tau\}$, and the *true frequency* $t_\pi(p)$ of p w.r.t. π is the probability that a transaction sampled from π contains p, that is,

$$t_\pi(p) = \Pr_{\tau \sim \pi}(p \sqsubseteq \tau).$$

In such a scenario, the final goal of the data mining process on $\mathcal{D}$ is to gain a better understanding of the process that generated the data, i.e., the distribution π, through the true frequencies of the patterns, which are unknown and only approximately reflected in the dataset $\mathcal{D}$.

5.2.2 Bound on the Maximum Deviation

Let us remember that under the assumptions of true frequent pattern mining (see Section 4.2.2), one is interested in finding good estimates for $t_\pi(p)$ simultaneously for each pattern $p \in \mathbb{U}$. In such a scenario, the true frequency $t_\pi(p)$ and the frequency $f_\mathcal{D}(p)$ of a pattern p on $\mathcal{D}$ represent, respectively, the expectation and the empirical average of a function associated with p. Thus, the maximum deviation (see Equation 2.1) is

$$\sup_{p \in \mathbb{U}} |t_\pi(p) - f_\mathcal{D}(p)|,$$

and one is interested in finding probabilistic upper bounds on such a measure, i.e., finding a $\mu \in (0,1)$ such that

$$\Pr\left(\sup_{p \in \mathbb{U}} |t_\pi(p) - f_\mathcal{D}(p)| \leq \mu \right) \geq 1 - \delta,$$

with a confidence parameter $\delta \in (0,1)$.

To find such a probabilistic upper bound, we now generalize the concepts based on the empirical VC-dimension introduced in Chapter 4 for sequential pattern mining to a general pattern mining task. First, we define the range space of patterns, generalizing Definition 10, and then we show how the empirical VC-dimension can be used to bound the maximum deviation in our scenario.

Definition 11. *Let $\mathbb{U}$ be a domain. We define $RS = (X, \mathcal{R})$ to be a range space associated with $\mathbb{U}$ such that:*

- *$X = \mathbb{U}$ is the set of patterns in $\mathbb{U}$;*
- *$\mathcal{R} = \{T(p) : p \in \mathbb{U}\}$ is a family of sets of transactions such that for each pattern p the set $T(p) = \{\tau \in \mathbb{U} : p \sqsubseteq \tau\}$ is the true support set of p.*

The following theorem is a generalization of the result for sequential patterns introduced in Chapter 4, Theorem 6. Here we provide it for a general pattern mining task.

Theorem 9. *Let $RS = (X, \mathcal{R})$ be the range space associated with $\mathbb{U}$, let π be a probability distribution on the domain $\mathbb{U}$, and let $\mathcal{D}$ be a finite bag of i.i.d. sample from π. If the empirical VC-dimension $EVC(RS, \mathcal{D})$ of RS on $\mathcal{D} \leq d$, then, given $\delta \in (0,1)$ and*

$$\mu = \sqrt{\frac{1}{2|\mathcal{D}|}\left(d + \ln\frac{1}{\delta}\right)},$$

$\sup_{p \in \mathbb{U}} |t_\pi(p) - f_{\mathcal{D}}(p)| \leq \mu$ with probability at least $1 - \delta$.

The proof is analogous to the proof of Theorem 6 in Section 4.2.2.

In Section 5.5 and 5.6, we discuss, respectively, an efficient computable upper bound on the empirical VC-dimension of sequential patterns and of itemsets, to bound the maximum deviation for these two data mining tasks.

5.3 Statistically Robust Pattern Mining

In this Section, we introduce the task of *statistically robust pattern (SRP) mining* from a *sequence of datasets*. Let us consider the scenario in which we have a sequence $\mathcal{D}_1^n = \{\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_n\}$ of n datasets, where each dataset $\mathcal{D}_i$ is a bag of $|\mathcal{D}_i|$ i.i.d. samples taken from a probability distribution π_i on $\mathbb{U}$, with $i \in \{1, \ldots, n\}$. Let $\Pi_1^n = \{\pi_1, \pi_2, \ldots, \pi_n\}$ denote the sequence of the n probability distributions and $\mathcal{T}_p = \{t_{\pi_1}(p), t_{\pi_2}(p), \ldots, t_{\pi_n}(p)\}$ the sequence of the true frequencies of the pattern p w.r.t. Π_1^n. In such a scenario, we are interested in finding patterns whose true frequencies w.r.t. Π_1^n respect a well defined *condition* $cond(\mathcal{T}_p)$ that describes the evolution of their true frequencies through the sequence. For example, one may be interested in finding patterns whose true frequencies are almost the same in all the probability distributions, or patterns whose true frequencies always increase/decrease, and so on. So, given n probabilities distribution $\Pi_1^n = \{\pi_1, \pi_2, \ldots, \pi_n\}$, a condition $cond(\mathcal{T}_p)$ on the true frequencies $\mathcal{T}_p$ that defines the patterns we are interested in, with $cond(\mathcal{T}_p) = 1$ when the condition is satisfied and

$cond(\mathcal{T}_p) = 0$ otherwise, *statistically robust pattern mining* is the task of reporting the set $SRP(\Pi_1^n)$ of all patterns whose true frequencies w.r.t Π_1^n respect $cond(\mathcal{T}_p)$, that is,

$$SRP(\Pi_1^n) = \{(p, \mathcal{T}_p) : p \in \mathbb{U} \wedge cond(\mathcal{T}_p) = 1\}.$$

Similarly to TFP mining, from a sequence of samples (the datasets $\mathcal{D}_1^n$) it is not possible to find the exact set $SRP(\Pi_1^n)$. Thus, one has to resort to approximations. Denoting by $\mathcal{F}_p = \{f_{\mathcal{D}_1}(p), f_{\mathcal{D}_2}(p), \ldots, f_{\mathcal{D}_n}(p)\}$ the sequence of the n frequencies of p in $\mathcal{D}_1^n$, we define a *false positives free (FPF) approximation* $\mathcal{A}_P$ of $SRP(\Pi_1^n)$ as

$$\mathcal{A}_P = \{(p, \mathcal{F}_p) : \exists (p, \mathcal{T}_p) \in SRP(\Pi_1^n)\}.$$

The approximation $\mathcal{A}_P$ does not contain *false positives*, that is, patterns $p \notin SRP(\Pi_1^n)$. In Section 5.4.4, we define an approximation that does not contain false negatives.

Now, we define three general types of patterns that can be described by the SRPs framework, and that we consider in the rest of this Chapter.

Emerging Patterns (EP): these are patterns whose true frequencies always increase over the sequence, i.e., patterns p for which $t_{\pi_{i+1}}(p) > t_{\pi_i}(p) + \varepsilon$, for all $i \in \{1, \ldots, n-1\}$, for some given *emerging threshold* $\varepsilon \in [0, 1)$. Formally, given an emerging threshold $\varepsilon \in [0, 1)$, we define the *emerging condition* $cond^E(\mathcal{T}_p)$ as

$$cond^E(\mathcal{T}_p) = \begin{cases} 1 & \text{if } t_{\pi_{i+1}}(p) > t_{\pi_i}(p) + \varepsilon, \forall i \in \{1, \ldots, n-1\} \\ 0 & \text{otherwise.} \end{cases} \tag{5.1}$$

Descending Patterns (DP): these are patterns p whose true frequencies always decrease over the sequence, i.e., patterns p for which $t_{\pi_i}(p) > t_{\pi_{i+1}}(p) + \varepsilon$ for all $i \in \{1, \ldots, n-1\}$, for some given emerging threshold $\varepsilon \in [0, 1)$. Formally, given an emerging threshold $\varepsilon \in [0, 1)$, we define the *descending condition* $cond^D(\mathcal{T}_p)$ as

$$cond^D(\mathcal{T}_p) = \begin{cases} 1 & \text{if } t_{\pi_i}(p) > t_{\pi_{i+1}}(p) + \varepsilon, \forall i \in \{1, \ldots, n-1\} \\ 0 & \text{otherwise.} \end{cases} \tag{5.2}$$

Stable Patterns (SP): these are patterns whose true frequencies in the n probability distributions are above a minimum frequency threshold θ and do not change too much. In particular, we consider patterns p for which $|t_{\pi_i}(p) - t_{\pi_j}(p)| \leq \alpha$ and $t_{\pi_i}(p) \geq \theta$ for all $i \neq j \in \{1, \ldots, n\}$, for some given *stability threshold* $\alpha \in (0, 1)$ and a minimum frequency threshold $\theta \in (0,1)$. Formally, given a stability threshold $\alpha \in (0,1)$ and minimum frequency threshold $\theta \in (0,1)$, we define the *stability condition* $cond^S(\mathcal{T}_p)$ as

$$cond^S(\mathcal{T}_p) = \begin{cases} 1 & \text{if } |t_{\pi_i}(p) - t_{\pi_j}(p)| \leq \alpha \wedge t_{\pi_i}(p) \geq \theta, \\ & \quad \forall\, i \neq j \in \{1,\dots,n\} \\ 0 & \text{otherwise.} \end{cases} \tag{5.3}$$

Let us note that many more types of patterns can be described by our proposed framework. For example, one may be interested in patterns whose true frequencies in the different probability distributions have a ratio larger than a user-defined constant, or may be interested in patterns whose true frequencies are stable in some distributions and then increase/decrease in others, or that first increase and then decrease, and so on. In addition, for the EP and DP tasks, we provided general conditions to describe such patterns, while one may also consider constraints using a minimum frequency threshold θ.

5.4 gRosSo: Approximating the SRPs

In this Section, we describe gRosSo, mininG statistically RObuSt patterns from a Sequence Of datasets, our strategy to provide a rigorous approximation of the SRPs. In particular, gRosSo aims to find an approximation that does not contain false positives (i.e., a FPF approximation, see Section 5.3) with high probability. In Sections 5.4.1-5.4.3 we show how to apply such a strategy to mine approximations of the three types of SRPs we defined in the previous Section. gRosSo can also be modified to find approximations with guarantees on the false negatives: in Section 5.4.4, we show how to use gRosSo to find such approximations for the EP task. Finally, in Section 5.4.5 we describe additional guarantees that can be obtained with gRosSo for both types of approximations.

Algorithm 6 shows the pseudo-code of gRosSo. For a fixed $cond(\mathcal{T}_p)$ that defines the SRPs we are interested in, and given the sequence $\mathcal{D}_1^n$ of n datasets and a confidence parameter $\delta \in (0,1)$ as inputs, we start computing an upper bound μ_i on the maximum deviation w.r.t π_i for each dataset $\mathcal{D}_i$, i.e., $\sup_{p\in\mathbb{U}} |t_{\pi_i}(p) - f_{\mathcal{D}_i}(p)| \leq \mu_i$, with $i \in \{1,\dots,n\}$ (line 2). Each upper bound is computed using confidence δ/n, thus $\Pr(\sup_{p\in\mathbb{U}} |t_{\pi_i}(p) - f_{\mathcal{D}_i}(p)| \leq \mu_i) \geq 1-\delta/n, \forall i \in \{1,\dots,n\}$. We denote by $\mu_1^n = \{\mu_1, \mu_2, \dots, \mu_n\}$ the sequence of the n upper bounds on the maximum deviations. (Such upper bounds can be computed, for example, using Theorem 9 and the VC-dimension.) Since $cond(\mathcal{T}_p)$ considers the true frequencies $\mathcal{T}_p$, which are unknown, we need to define a new condition $cond_P(\mathcal{F}_p, \mu_1^n)$ on

Algorithm 6: gRosSo: find a FPF approximation $\mathcal{A}_P$ of $SRP(\Pi_1^n)$.

```
   Data: Datasets $\mathcal{D}_1^n$, $\delta \in (0,1)$.
   Result: Set $\mathcal{A}_P$ that is a FPF approx. of $SRP(\Pi_1^n)$ with probability $\geq 1-\delta$.
1  foreach $\mathcal{D}_i \in \mathcal{D}_1^n$ do
2      $\mu_i \leftarrow$ ComputeMaxDev($\mathcal{D}_i, \delta/n$);
3      $\tilde{\theta}_i \leftarrow$ min. frequency threshold for $\mathcal{D}_i$ computed considering $cond_P(\mathcal{F}_p, \mu_i^n)$;
4  $\mathcal{B} \leftarrow FP(\mathcal{D}_k, \tilde{\theta}_k)$, with $k = \arg\max_{i\in\{1,\ldots,n\}} \tilde{\theta}_i$;
5  $\mathcal{A}_P \leftarrow \emptyset$;
6  foreach $(p, f_{\mathcal{D}_k}(p)) \in \mathcal{B}$ do
7      $\mathcal{F}_p \leftarrow$ empty array of n elements;
8      $\mathcal{F}_p[k] \leftarrow f_{\mathcal{D}_k}(p)$; /* $\mathcal{F}_p[k]$: $k$-th element of $\mathcal{F}_p$ */
9      $\mathcal{A}_P \leftarrow \mathcal{A}_P \cup (p, \mathcal{F}_p)$;
10 foreach $\mathcal{D}_i \in \mathcal{D}_1^n \setminus \mathcal{D}_k$ do
11     foreach $(p, \mathcal{F}_p) \in \mathcal{A}_P$ do
12         $\mathcal{F}_p[i] \leftarrow$ ComputeFrequency($\mathcal{D}_i, p$);
13         if $cond_P(\mathcal{F}_p, \mu_1^n) = 0$ then
14             $\mathcal{A}_P \leftarrow \mathcal{A}_P \setminus (p, \mathcal{F}_p)$;
15 return $\mathcal{A}_P$;
```

the frequencies $\mathcal{F}_p$ and on the upper bounds μ_1^n. Such a new FPF condition takes into account the uncertainty of the data in our samples, i.e., the datasets, in order to avoid false positives, and, for a pattern p, it must be $cond(\mathcal{T}_p) = 0 \implies cond_P(\mathcal{F}_p, \mu_1^n) = 0$ if $\sup_{p\in\mathbb{U}} |t_{\pi_i}(p) - f_{\mathcal{D}_i}(p)| \leq \mu_i$ holds $\forall i \in \{1, \ldots, n\}$. Figure 5.1 shows such conditions for the EP and SP scenarios as examples. Let us note that $cond_P(\mathcal{F}_p, \mu_1^n)$ can be also evaluated only considering a subsequence of the frequencies $\mathcal{F}_p$, and if $cond_P(\mathcal{F}_p, \mu_1^n) = 0$ for some subsequence, then there are no guarantees that $cond(\mathcal{T}_p) = 1$. Then, we aim to find a starting set of possible candidates. For each dataset $\mathcal{D}_i$, we compute the minimum frequency threshold $\tilde{\theta}_i$ which the patterns must have in such a dataset to verify $cond_P(\mathcal{F}_p, \mu_1^n)$ (line 3). We then mine the dataset $\mathcal{D}_k$, where $k = \arg\max_{i\in\{1,\ldots,n\}} \tilde{\theta}_i$, with the corresponding minimum frequency threshold $\tilde{\theta}_k$, obtaining the set $\mathcal{B} = FP(\mathcal{D}_k, \tilde{\theta}_k)$ of the starting candidates (line 4). The idea is to mine the dataset with the highest minimum frequency threshold in order to obtain a set of possible candidates that is as small as possible. Finally, we explore the remaining datasets. For each $\mathcal{D}_i \in \mathcal{D}_1^n \setminus \mathcal{D}_k$ and for each $p \in \mathcal{B}$, we compute its frequency $f_{\mathcal{D}_i}(p)$ in $\mathcal{D}_i$ (line 12) and check whether $cond_P(\mathcal{F}_p, \mu_1^n) = 1$, considering the subsequence of the frequencies $\mathcal{F}_p$ that has already been computed. If $cond_P(\mathcal{F}_p, \mu_1^n) = 0$, there are no guarantees that $cond(\mathcal{T}_p) = 1$, and we remove such a pattern from the set of the possible candidates (lines 13-14). Then,

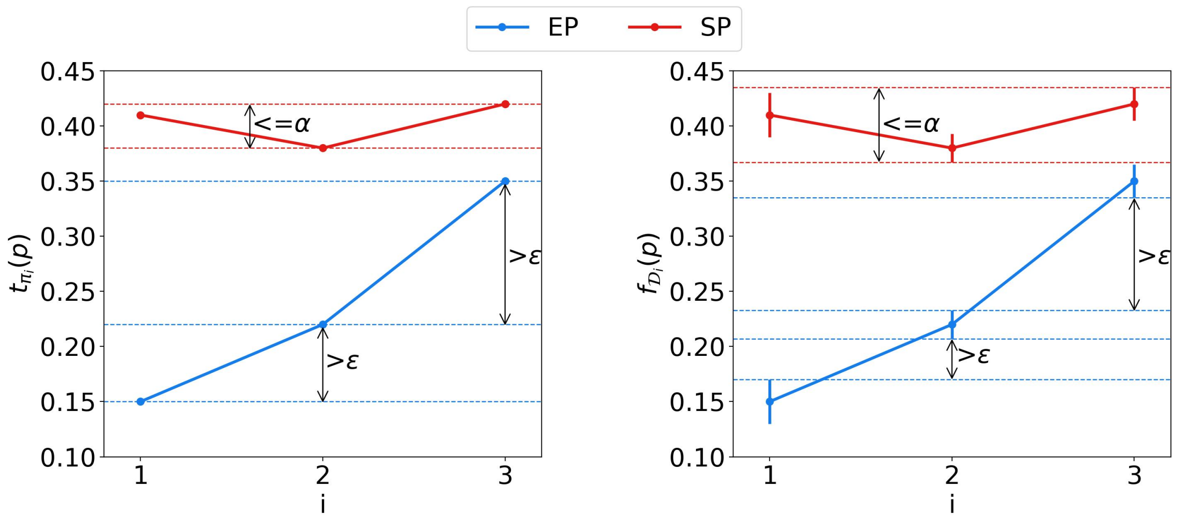

Fig. 5.1 FPF conditions for the EP and SP. The two panels show the $cond(\mathcal{T}_p)$ (left) and the corresponding $cond_P(\mathcal{F}_p, \mu_1^n)$ (right), which takes into account the uncertainty of the data and avoids false positives, for the EP and SP tasks, with $n = 3$ datasets.

the output are the patterns that have not been removed from the set of the possible candidates (line 15).

Theorem 10. *The set $\mathcal{A}_P$ returned by* gRosSo *is a FPF approximation of $SRP(\Pi_1^n)$ with probability $\geq 1 - \delta$.*

Proof. From the definition of $cond_P(\mathcal{F}_p, \mu_1^n)$, $cond(\mathcal{T}_p) = 0 \implies cond_P(\mathcal{F}_p, \mu_1^n) = 0$ if $\sup_{p \in \mathbb{U}} |t_{\pi_i}(p) - f_{\mathcal{D}_i}(p)| \leq \mu_i$ holds $\forall\, i \in \{1, \ldots, n\}$. In such a scenario, only the patterns $p \in SRP(\Pi_1^n)$ can appear in $\mathcal{A}_P$, and thus $\mathcal{A}_P$ is a FPF approximation of $SRP(\Pi_1^n)$. Now, let us define the event E_i as the event in which $\sup_{p \in \mathbb{U}} |t_{\pi_i}(p) - f_{\mathcal{D}_i}(p)| > \mu_i$, with $i \in \{1, \ldots, n\}$. From the choice of the confidence parameter used to compute the upper bounds on the maximum deviation, we know that $\Pr(E_i) < \delta/n$. So, we have $\Pr(\exists i \in \{1, \ldots, n\} : \sup_{p \in \mathbb{U}} |t_{\pi_i}(p) - f_{\mathcal{D}_i}(p)| > \mu_i) = \Pr(\cup_{i=1}^n E_i) \leq \sum_{i=1}^n \Pr(E_i) < \delta$. Thus, the set $\mathcal{A}_P$ returned by gRosSo is a FPF approximation of $SRP(\Pi_1^n)$ with probability $\geq 1 - \delta$, which concludes the proof. $\square$

5.4.1 FPF Approximation of the EP

Now, we apply the strategy defined above to find a FPF approximation of the EP. Starting from $cond^E(\mathcal{T}_p)$ (Equation 5.1), we define $cond^E_P(\mathcal{F}_p, \mu_1^n)$ as

$$cond^E_P(\mathcal{F}_p, \mu_1^n) = \begin{cases} 1 & \text{if } f_{\mathcal{D}_{i+1}}(p) - \mu_{i+1} - (f_{\mathcal{D}_i}(p) + \mu_i) > \varepsilon, \\ & \quad \forall\, i \in \{1, \dots, n-1\} \\ 0 & \text{otherwise.} \end{cases}$$

For a given $i \in \{1, \dots, n-1\}$, such a condition represents the scenario in which $t_{\pi_{i+1}}(p)$ and $t_{\pi_i}(p)$ assume the values $f_{\mathcal{D}_{i+1}}(p) - \mu_{i+1}$ and $f_{\mathcal{D}_i}(p) + \mu_i$, respectively, that are the values at which their distance is minimum over all possible values that they can assume. Only if such a condition is true, we are guaranteed that $t_{\pi_{i+1}}(p) > t_{\pi_i}(p) + \varepsilon$. (See Figure 5.1.) Then, starting from such a condition, we compute the minimum frequency threshold for each dataset. Since it must be $f_{\mathcal{D}_2}(p) - \mu_2 > f_{\mathcal{D}_1}(p) + \mu_1 + \varepsilon$ and $f_{\mathcal{D}_3}(p) - \mu_3 > f_{\mathcal{D}_2}(p) + \mu_2 + \varepsilon$, and thus $f_{\mathcal{D}_3}(p) > f_{\mathcal{D}_1}(p) + 2 \cdot \varepsilon + \mu_1 + \mu_3 + 2 \cdot \mu_2$, iterating such a reasoning for all the n datasets and considering $f_{\mathcal{D}_1}(p) \geq 0$, we obtain the minimum frequency threshold $\tilde{\theta}^E_n$ for the dataset $\mathcal{D}_n$,

$$\tilde{\theta}^E_n = (n-1) \cdot \varepsilon + \mu_1 + \mu_n + \sum_{i=2}^{n-1} 2 \cdot \mu_i,$$

the highest over all the n datasets. Thus, the set $FP(\mathcal{D}_n, \tilde{\theta}^E_n)$ provides the starting candidates. Finally, starting from $\mathcal{D}_{n-1}$ and ending with $\mathcal{D}_1$, we analyze the remaining datasets and check whether the candidates verify $cond^E_P(\mathcal{F}_p, \mu_1^n)$.

Theorem 11. *$cond^E(\mathcal{T}_p) = 0 \implies cond^E_P(\mathcal{F}_p, \mu_1^n) = 0$.*

Proof. Let us consider that $\sup_{p \in \mathbb{U}} |t_{\pi_i}(p) - f_{\mathcal{D}_i}(p)| \leq \mu_i, \forall\, i \in \{1, \dots, n\}$. Thus, we have that for all patterns $p \in \mathbb{U}$, it results $t_{\pi_i}(p) \in [f_{\mathcal{D}_i}(p) - \mu_i, f_{\mathcal{D}_i}(p) + \mu_i]$, $\forall\, i \in \{1, \dots, n\}$. Let p' be a pattern s.t. $cond^E(\mathcal{T}_{p'}) = 0$. From Equation 5.1, there is at least a couple of consecutive distribution π_j, π_{j+1}, with $j \in \{1, \dots, n-1\}$, s.t. $t_{\pi_{j+1}}(p') \leq t_{\pi_j}(p') + \varepsilon$. Since we know that $t_{\pi_{j+1}}(p') \in [f_{\mathcal{D}_{j+1}}(p') - \mu_{j+1}, f_{\mathcal{D}_{j+1}}(p') + \mu_{j+1}]$ and that $t_{\pi_j}(p') \in [f_{\mathcal{D}_j}(p') - \mu_j, f_{\mathcal{D}_j}(p') + \mu_j]$, the condition $f_{\mathcal{D}_{j+1}}(p') - \mu_{j+1} - (f_{\mathcal{D}_j}(p') + \mu_j) > \varepsilon$, cannot be verified for such p', and thus $cond^E_P(\mathcal{F}_{p'}, \mu_1^n) = 0$, which concludes the proof. □

If one is interested in patterns with a true frequency above a value $\theta \in (0, 1)$, i.e., $t_{\pi_i}(p) \geq \theta$, $\forall\, i \in \{1, \dots, n\}$, the following strategy can be used to reduce the set of starting candidates. Since we require that $f_{\mathcal{D}_1}(p) \geq \theta + \mu_1$ to discard possible false positives, a factor $\theta + \mu_1$ must be added to $\tilde{\theta}_n^E$. Instead, if one is interested in patterns p with $t_{\pi_n}(p) \geq \theta$, the minimum frequency threshold $\tilde{\theta}_n^E$ for dataset $\mathcal{D}_n$ is

$$\tilde{\theta}_n^E = \max\{(n-1)\cdot\varepsilon + \mu_1 + \mu_n + \sum_{i=2}^{n-1} 2\cdot\mu_i,\ \theta + \mu_n\}.$$

5.4.2 FPF Approximation of the DP

Using the same approach proposed to approximate the EP, it is possible to approximate the DP. Starting from $cond^D(\mathcal{T}_p)$ (Equation 5.2), we define $cond_P^D(\mathcal{F}_p, \mu_1^n)$ as

$$cond_P^D(\mathcal{F}_p, \mu_1^n) = \begin{cases} 1 & \text{if } f_{\mathcal{D}_i}(p) - \mu_i - (f_{\mathcal{D}_{i+1}}(p) + \mu_{i+1}) > \varepsilon, \\ & \quad \forall\, i \in \{1, \dots, n-1\} \\ 0 & \text{otherwise.} \end{cases}$$

Iterating such a condition for all the n datasets, we obtain the minimum frequency threshold $\tilde{\theta}_1^D = \tilde{\theta}_n^E$ for the dataset $\mathcal{D}_1$, that is the highest over all the n datasets. Thus, the set $FP(\mathcal{D}_1, \tilde{\theta}_1^D)$ provides the starting candidates. Finally, starting from $\mathcal{D}_2$ and ending with $\mathcal{D}_n$, we analyze the remaining datasets and check whether the candidates verify $cond_P^D(\mathcal{F}_p, \mu_1^n)$. In the case of a minimum frequency threshold $\theta \in (0,1)$, reasoning analogous to the EP can be applied.

Theorem 12. $cond^D(\mathcal{T}_p) = 0 \implies cond_P^D(\mathcal{F}_p, \mu_1^n) = 0.$

The proof is analogous to the proof of Theorem 11.

5.4.3 FPF Approximation of the SP

Finally, we apply the strategy defined above to find an approximation of the SP. Starting from $cond^S(\mathcal{T}_p)$ (Equation 5.3), we define $cond_P^S(\mathcal{F}_p, \mu_1^n)$ as

$$cond_P^S(\mathcal{F}_p, \mu_1^n) = \begin{cases} 1 & \text{if } f_{\mathcal{D}_i}(p) + \mu_i - (f_{\mathcal{D}_j}(p) - \mu_j) \leq \alpha \\ & \quad \wedge\, f_{\mathcal{D}_j}(p) + \mu_j - (f_{\mathcal{D}_i}(p) - \mu_i) \leq \alpha, \\ & \quad \wedge\, f_{\mathcal{D}_i}(p) - \mu_i \geq \theta, \\ & \quad \forall\, i \neq j \in \{1, \dots, n\} \\ 0 & \text{otherwise.} \end{cases}$$

Given $i \neq j \in \{1, \dots, n\}$, the first two conditions represent the scenario in which $t_{\pi_i}(p)$ and $t_{\pi_j}(p)$ assume the values $f_{\mathcal{D}_i}(p) - \mu_i$ and $f_{\mathcal{D}_j}(p) + \mu_j$, respectively, if $f_{\mathcal{D}_i}(p) < f_{\mathcal{D}_j}(p)$, or respectively the values $f_{\mathcal{D}_j}(p)-\mu_j$ and $f_{\mathcal{D}_i}(p)+\mu_i$ if $f_{\mathcal{D}_j}(p) < f_{\mathcal{D}_i}(p)$, that are the values at which their distance is maximum over all possible values that they can assume. Only if such conditions are true, we can prove that $|t_{\pi_i}(p) - t_{\pi_j}(p)| \leq \alpha$. The third condition, instead, represents the scenario in which $t_{\pi_i}(p)$ assumes the value $f_{\mathcal{D}_i}(p) - \mu_i$, that is the minimum value that it can assume. Only if such a condition is true, we can prove that $t_{\pi_i}(p) \geq \theta$. (See Figure 5.1.) The only condition that affects the minimum frequency thresholds $\tilde{\theta}_i^S$ is $f_{\mathcal{D}_i}(p) \geq \theta+\mu_i$, $\forall i \in \{1,\dots,n\}$. So, we have $\tilde{\theta}_i^S = \theta+\mu_i$, $\forall i \in \{1,\dots,n\}$, and the set $FP(\mathcal{D}_k, \tilde{\theta}_k^S)$, with $k = \arg\max_{i\in\{1,\dots,n\}} \tilde{\theta}_i^S$, provides the starting candidates. Finally, we analyze the remaining datasets $\mathcal{D}_i \in \mathcal{D}_1^n \backslash \mathcal{D}_k$ and check whether the candidates verify $cond_P^S(\mathcal{F}_p, \mu_1^n)$.

Theorem 13. *$cond^S(\mathcal{T}_p) = 0 \implies cond_P^S(\mathcal{F}_p, \mu_1^n) = 0$.*

Proof. Let us consider that $\sup_{p\in\mathbb{U}} |t_{\pi_i}(p) - f_{\mathcal{D}_i}(p)| \leq \mu_i$, $\forall\, i \in \{1,\dots,n\}$. Thus, we have that for all patterns $p \in \mathbb{U}$, it results $t_{\pi_i}(p) \in [f_{\mathcal{D}_i}(p) - \mu_i, f_{\mathcal{D}_i}(p) + \mu_i]$, $\forall\, i \in \{1,\dots,n\}$. Let p' be a pattern s.t. $cond^S(\mathcal{T}_{p'}) = 0$. From Equation 5.3, there is at least a distribution π_i, with $i \in \{1,\dots,n\}$ s.t. $t_{\pi_i}(p') < \theta$ and/or there is at least a couple of distributions π_k, π_j, with $k \neq j \in \{1,\dots,n\}$, s.t. $|t_{\pi_j}(p') - t_{\pi_k}(p')| > \alpha$. First, let us consider the case in which there is a distribution π_i, with $i \in \{1,\dots,n\}$, s.t. $t_{\pi_i}(p') < \theta$. Since we know that $t_{\pi_i}(p') \in [f_{\mathcal{D}_i}(p') - \mu_i, f_{\mathcal{D}_i}(p') + \mu_i]$, the condition $f_{\mathcal{D}_i}(p') - \mu_i \geq \theta$ cannot be verified, and thus $cond_P^S(\mathcal{F}_{p'}, \mu_1^n) = 0$. Now, let us consider the case in which there is a couple of distributions π_k, π_j, with $k \neq j \in \{1,\dots,n\}$ s.t. $|t_{\pi_j}(p') - t_{\pi_k}(p')| > \alpha$. Since we know that $t_{\pi_j}(p') \in [f_{\mathcal{D}_j}(p')-\mu_j, f_{\mathcal{D}_j}(p')+\mu_j]$ and that $t_{\pi_k}(p') \in [f_{\mathcal{D}_k}(p')-\mu_k, f_{\mathcal{D}_k}(p')+\mu_k]$, the condition $f_{\mathcal{D}_j}(p') + \mu_j - (f_{\mathcal{D}_k}(p') - \mu_k) \leq \alpha$ cannot be verified, if $f_{\mathcal{D}_j}(p') > f_{\mathcal{D}_k}(p')$, while the condition $f_{\mathcal{D}_k}(p') + \mu_k - (f_{\mathcal{D}_j}(p') - \mu_j) \leq \alpha$ cannot be verified if $f_{\mathcal{D}_j}(p') < f_{\mathcal{D}_k}(p')$, and thus $cond_P^S(\mathcal{F}_{p'}, \mu_1^n) = 0$, which concludes the proof. □

5.4.4 Guarantees on False Negatives

In this Section, we explain how gRosSo can be modified to obtain an approximation without false negatives with high probability. Analogously to what done in Section 5.3, we first define a *false negatives free (FNF) approximation* $\mathcal{A}_N$ of $SRP(\Pi_1^n)$ as

$$\mathcal{A}_N = \{(p, \mathcal{F}_p), \forall (p, \mathcal{T}_p) \in SRP(\Pi_1^n)\}.$$

The approximation $\mathcal{A}_N$ does not contain *false negatives*, that is, it contains all patterns $p \in SRP(\Pi_1^n)$.

Our algorithm gRosSo can be used to obtain FNF approximations. The procedure is the same described above (see Algorithm 6) but we need to define a new condition

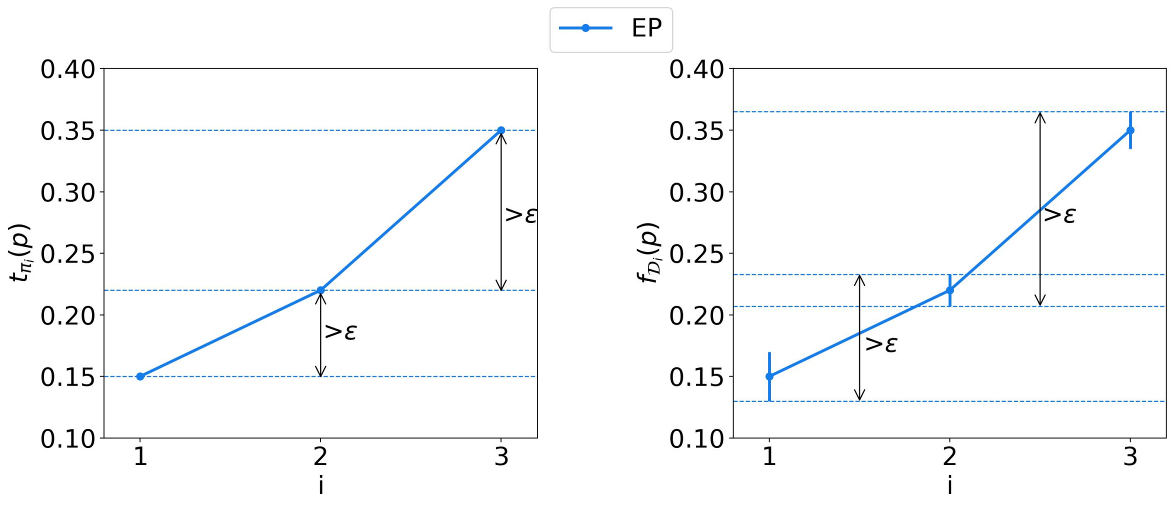

Fig. 5.2 FNF condition for the EP. The two panels show the $cond(\mathcal{T}_p)$ (left) and the corresponding $cond_N(\mathcal{F}_p, \mu_1^n)$ (right), which takes into account the uncertainty of the data and avoids false negatives, for the EP task, with $n = 3$ datasets.

$cond_N(\mathcal{F}_p, \mu_1^n)$, a FNF condition that takes into account the uncertainty in the data in order to avoid false negatives, and for a pattern p it must be $cond(\mathcal{T}_p) = 1 \implies cond_N(\mathcal{F}_p, \mu_1^n) = 1$ if $\sup_{p \in \mathbb{U}} |t_{\pi_i}(p) - f_{\mathcal{D}_i}(p)| \leq \mu_i$ holds $\forall i \in \{1, \ldots, n\}$. Figure 5.2 shows such a condition for the EP scenario as an example. Then, we compute the minimum frequency threshold $\hat{\theta}_i$ for each dataset $\mathcal{D}_i$, with $i \in \{1, \ldots, n\}$, considering $cond_N(\mathcal{F}_p, \mu_i^n)$, and we mine the set of the starting candidates from the dataset with the highest minimum frequency threshold.

Theorem 14. *The set $\mathcal{A}_N$ returned by* GRoSSo *using $cond_N(\mathcal{F}_p, \mu_1^n)$ is a FNF approximation of $SRP(\Pi_1^n)$ with probability $\geq 1 - \delta$.*

Proof. From the definition of $cond_N(\mathcal{F}_p, \mu_1^n)$, $cond(\mathcal{T}_p) = 1 \implies cond_N(\mathcal{F}_p, \mu_1^n) = 1$ if $\sup_{p \in \mathbb{U}} |t_{\pi_i}(p) - f_{\mathcal{D}_i}(p)| \leq \mu_i$ holds $\forall\, i \in \{1, \ldots, n\}$. In such a scenario, all the patterns $p \in SRP(\Pi_1^n)$ appear in $\mathcal{A}_N$, and thus $\mathcal{A}_N$ is a FNF approximation of $SRP(\Pi_1^n)$. The remaining of the proof is analogous of the proof of Theorem 10. □

FNF Approximation of the EP

Now, we apply the strategy defined above to find a FNF approximation of the EP. With a similar reasoning, it is possible to mine a FNF approximation of the descending and stable patterns. Starting from $cond^E(\mathcal{T}_p)$ (Equation 5.1), we define $cond^E_N(\mathcal{F}_p, \mu_1^n)$ as

$$cond^E_N(\mathcal{F}_p, \mu_1^n) = \begin{cases} 1 & \text{if } f_{\mathcal{D}_{i+1}}(p) + \mu_{i+1} - (f_{\mathcal{D}_i}(p) - \mu_i) > \varepsilon, \\ & \quad \forall\, i \in \{1, \dots, n-1\} \\ 0 & \text{otherwise.} \end{cases}$$

For a given $i \in \{1, \dots, n-1\}$, such a condition represents the scenario in which $t_{\pi_{i+1}}(p)$ and $t_{\pi_i}(p)$ assume the values $f_{\mathcal{D}_{i+1}}(p) + \mu_{i+1}$ and $f_{\mathcal{D}_i}(p) - \mu_i$, respectively, that are the values at which their distance is maximum over all possible values that they can assume. Only if such a condition is false, we can prove that $t_{\pi_{i+1}}(p) \leq t_{\pi_i}(p) + \varepsilon$. (See Figure 5.2.) Then, starting from such a condition, we compute the minimum frequency threshold for each dataset. Since it must be $f_{\mathcal{D}_2}(p) + \mu_2 > f_{\mathcal{D}_1}(p) - \mu_1 + \varepsilon$ and $f_{\mathcal{D}_3}(p) + \mu_3 > f_{\mathcal{D}_2}(p) - \mu_2 + \varepsilon$, and thus $f_{\mathcal{D}_3}(p) > f_{\mathcal{D}_1}(p) + 2 \cdot \varepsilon - \mu_1 - \mu_3 - 2 \cdot \mu_2$, iterating such a reasoning for all the n datasets and considering $f_{\mathcal{D}_1}(p) \geq 0$, we obtain the minimum frequency threshold

$$\hat{\theta}_i^E = (i-1) \cdot \varepsilon - \mu_1 - \mu_i - \sum_{j=2}^{i-1} 2 \cdot \mu_j,$$

for each dataset $\mathcal{D}_i$, $i \in \{2, \dots, n\}$, and $\hat{\theta}_1^E = 0$. Thus, the set $FP(\mathcal{D}_k, \hat{\theta}_k^E)$, with $k = \arg\max_{i \in \{1,\dots,n\}} \hat{\theta}_i^E$, provides the starting candidates. Then, we analyze the remaining datasets $\mathcal{D}_i \in \mathcal{D}_1^n \setminus \mathcal{D}_k$ and check whether the candidates verify $cond^E_N(\mathcal{F}_p, \mu_1^n)$. Let us note that, depending on the values of ε and μ_1^n, the highest minimum frequency threshold $\hat{\theta}_k^E$ can be equal or very close to 0, resulting in a huge amount of starting candidates, sometimes infeasible to mine. Thus, to obtain FNF approximations, a reasonable solution is to only consider patterns with a minimum true frequency. If one is interested in patterns with a true frequency above a value $\theta \in (0,1)$, i.e., $t_{\pi_i}(p) \geq \theta$, $\forall\, i \in \{1, \dots, n\}$, the following strategy can be used to reduce the set of starting candidates. Since we require that $f_{\mathcal{D}_i}(p) \geq \theta - \mu_i$ for all $i \in \{1, \dots, n\}$ to discard possible false negatives, we obtain the minimum frequency threshold

$$\hat{\theta}_i^E = \max\{(i-1) \cdot \varepsilon + \theta - \mu_i - \sum_{j=1}^{i-1} 2 \cdot \mu_j, \theta - \mu_i\},$$

for each dataset $\mathcal{D}_i$, $i \in \{1, \dots, n\}$. Instead, if one is interested in patterns p with $t_{\pi_n}(p) \geq \theta$, the highest minimum frequency threshold is the maximum between $\theta - \mu_n$, for dataset $\mathcal{D}_n$, and the minimum frequency threshold described above without frequency constraints.

Theorem 15. *$cond^E(\mathcal{T}_p) = 1 \implies cond^E_N(\mathcal{F}_p, \mu_1^n) = 1$.*

Proof. Let us consider that $\sup_{p\in\mathbb{U}} |t_{\pi_i}(p) - f_{\mathcal{D}_i}(p)| \leq \mu_i$, $\forall\, i \in \{1,\ldots,n\}$. Thus, we have that for all patterns $p \in \mathbb{U}$, it results $t_{\pi_i}(p) \in [f_{\mathcal{D}_i}(p) - \mu_i, f_{\mathcal{D}_i}(p) + \mu_i]$, $\forall\, i \in \{1,\ldots,n\}$. Let p' be a pattern s.t. $cond^E(\mathcal{T}_{p'}) = 1$. From Equation 5.1, we have $t_{\pi_{j+1}}(p') > t_{\pi_j}(p') + \varepsilon$ for all couples of consecutive distributions π_j, π_{j+1}, with $j \in \{1,\ldots,n-1\}$. Since we know that $t_{\pi_{j+1}}(p') \in [f_{\mathcal{D}_{j+1}}(p') - \mu_{j+1}, f_{\mathcal{D}_{j+1}}(p') + \mu_{j+1}]$ and that $t_{\pi_j}(p') \in [f_{\mathcal{D}_j}(p') - \mu_j, f_{\mathcal{D}_j}(p') + \mu_j]$, the condition $f_{\mathcal{D}_{j+1}}(p') + \mu_{j+1} - (f_{\mathcal{D}_j}(p') - \mu_j) > \varepsilon$ is verified for such p' for all $j \in \{1,\ldots,n-1\}$, and thus $cond^E_N(\mathcal{F}_{p'}, \mu_1^n) = 1$, which concludes the proof. □

5.4.5 Additional Guarantees of gRosSo

In this Section, we provide additional guarantees of gRosSo for both types of approximations. In particular, it is possible to derive guarantees on the false negatives that can appear in a FPF approximation returned by gRosSo, and, vice versa, guarantees on the false positives that can appear in a FNF approximation. Such guarantees differ considering different types of patterns (i.e., emerging, descending, and stable), and thus, they must be separately derived for each type of pattern. Here, we prove additional guarantees for the emerging patterns but, with a similar reasoning, it is possible to obtain analogous guarantees for the descending and stable patterns.

Theorem 16. *For any pattern p with $t_{\pi_{i+1}}(p) > t_{\pi_i}(p) + \varepsilon + 2\cdot\mu_i + 2\cdot\mu_{i+1}$ $\forall i \in \{1,\ldots,n-1\}$, $cond^E_P(\mathcal{F}_p, \mu_1^n) = 1$.*

Proof. Let us consider that $\sup_{p\in\mathbb{U}} |t_{\pi_i}(p) - f_{\mathcal{D}_i}(p)| \leq \mu_i$, $\forall\, i \in \{1,\ldots,n\}$. Thus, we have that for all patterns $p \in \mathbb{U}$, it results $t_{\pi_i}(p) \in [f_{\mathcal{D}_i}(p) - \mu_i, f_{\mathcal{D}_i}(p) + \mu_i]$, $\forall i \in \{1,\ldots,n\}$. Let p' be a pattern s.t. $t_{\pi_{i+1}}(p') > t_{\pi_i}(p') + \varepsilon + 2\cdot\mu_i + 2\cdot\mu_{i+1}$ $\forall i \in \{1,\ldots,n-1\}$. Since we know that $t_{\pi_{i+1}}(p') \in [f_{\mathcal{D}_{i+1}}(p') - \mu_{i+1}, f_{\mathcal{D}_{i+1}}(p') + \mu_{i+1}]$ and that $t_{\pi_i}(p') \in [f_{\mathcal{D}_i}(p') - \mu_i, f_{\mathcal{D}_i}(p') + \mu_i]$ $\forall i \in \{1,\ldots,n-1\}$, then we have that $f_{\mathcal{D}_{i+1}}(p') - \mu_{i+1} > f_{\mathcal{D}_i}(p') + \mu_i + \varepsilon$ $\forall i \in \{1,\ldots,n-1\}$, and thus $cond^E_P(\mathcal{F}_{p'}, \mu_1^n) = 1$, which concludes the proof. □

Theorem 17. *For any pattern p with $t_{\pi_{i+1}}(p) \leq t_{\pi_i}(p) + \varepsilon - 2\cdot\mu_i - 2\cdot\mu_{i+1}$ $\forall i \in \{1,\ldots,n-1\}$, $cond^E_N(\mathcal{F}_p, \mu_1^n) = 0$.*

Proof. Let us consider that $\sup_{p\in\mathbb{U}} |t_{\pi_i}(p) - f_{\mathcal{D}_i}(p)| \leq \mu_i$, $\forall\, i \in \{1,\ldots,n\}$. Thus, we have that for all patterns $p \in \mathbb{U}$, it results $t_{\pi_i}(p) \in [f_{\mathcal{D}_i}(p) - \mu_i, f_{\mathcal{D}_i}(p) + \mu_i]$, $\forall i \in \{1,\ldots,n\}$. Let p' be a pattern s.t. $t_{\pi_{i+1}}(p') \leq t_{\pi_i}(p') + \varepsilon - 2\cdot\mu_i - 2\cdot\mu_{i+1}$ $\forall i \in \{1,\ldots,n-1\}$. Since we know that $t_{\pi_{i+1}}(p') \in [f_{\mathcal{D}_{i+1}}(p') - \mu_{i+1}, f_{\mathcal{D}_{i+1}}(p') + \mu_{i+1}]$ and that $t_{\pi_i}(p') \in [f_{\mathcal{D}_i}(p') - \mu_i, f_{\mathcal{D}_i}(p') + \mu_i]$ $\forall i \in \{1,\ldots,n-1\}$, then we have that $f_{\mathcal{D}_{i+1}}(p') + \mu_{i+1} \leq f_{\mathcal{D}_i}(p') - \mu_i + \varepsilon$ and thus $cond^E_N(\mathcal{F}_{p'}, \mu_1^n) = 0$, which concludes the proof. □

Theorems 16 and 17 provide additional guarantees for the emerging patterns returned by gRosSo. In particular, Theorem 16 provides additional guarantees for a FPF approximation returned by gRosSo, stating that a pattern p with $t_{\pi_{i+1}}(p) > t_{\pi_i}(p) + \varepsilon + 2 \cdot \mu_i + 2 \cdot \mu_{i+1}$ $\forall i \in \{1, \dots, n-1\}$ is certainly included in a FPF approximation. Instead, Theorem 17 provides additional guarantees for a FNF approximation returned by gRosSo, stating that a pattern p with $t_{\pi_{i+1}}(p) \leq t_{\pi_i}(p) + \varepsilon - 2 \cdot \mu_i - 2 \cdot \mu_{i+1}$ $\forall i \in \{1, \dots, n-1\}$ can not appear in a FNF approximation.

5.5 Mining Statistically Robust Sequential Patterns

In this Section, we refresh the task of sequential pattern mining (see Section 2.1.2), as a concrete realization of the general framework of pattern mining we introduced in Section 5.2.1. Then, we introduce a novel algorithm to compute an upper bound on the capacity of a sequence which can be used to compute an upper bound on the empirical VC-dimension of sequential patterns. Finally, we discuss a VC-dimension based strategy to bound the maximum deviation of the true frequencies of sequential patterns, which can be used in the SRP mining scenario.

Let us remember, from Section 2.1.2, that a *sequential pattern*, or *sequence*, $s = \langle S_1, S_2, \dots, S_k \rangle$ is a finite ordered sequence of *itemsets* S_i, with $i \in \{1, \dots, k\}$, which are sets of elements called *items*. The *length* $|s|$ of s is the number of itemsets in s, the *item-length* $||s||$ of s is the sum of the sizes of the itemsets in s, that is,

$$||s|| = \sum_{i=1}^{|s|} |S_i|.$$

A sequential pattern $y = \langle Y_1, Y_2, \dots, Y_a \rangle$ is a *subsequence* of an other sequential pattern $w = \langle W_1, W_2, ..., W_b \rangle$, denoted by $y \sqsubseteq w$, if and only if there exists a sequence of naturals $1 \leq i_1 < i_2 < \cdots < i_a \leq b$ such that $Y_1 \subseteq W_{i_1}, Y_2 \subseteq W_{i_2}, \dots, Y_a \subseteq W_{i_a}$. The *capacity* $c(s)$ of a sequence s is the number of distinct subsequences of s, i.e., $c(s) = |\{a : a \sqsubseteq s\}|$. Finally, the *domain* $\mathbb{S}$ is the set of all the sequences which can be built with itemsets that are subsets of a given set of items $\mathcal{I}$.

Given a dataset $\mathcal{D}$ for the sequential pattern mining task, that is a finite bag of transactions sampled from $\mathbb{S}$ in accordance with π, we aim to compute the empirical VC-dimension $EVC(RS, \mathcal{D})$ of the range space (see Definition 11) associated with $\mathbb{S}$ on the dataset $\mathcal{D}$ in order to find a probabilistic bound $\mu \in (0, 1)$ on the maximum deviation $\sup_{s \in \mathbb{S}} |t_\pi(s) - f_\mathcal{D}(s)|$. In particular, given $EVC(RS, \mathcal{D})$ and using Theorem 9, it is possible to compute a $\mu \in (0, 1)$ s.t. $\sup_{s \in \mathbb{S}} |t_\pi(s) - f_\mathcal{D}(s)| \leq \mu$.

Let us remember, from Section 4.2.2 of Chapter 4, that the exact computation of the empirical VC-dimension $EVC(RS, \mathcal{D})$ of sequential patterns on the dataset $\mathcal{D}$ is computationally expensive. The s-index (see Definition 6) introduced by Servan-Schreiber et al. [75] and the s-bound (see Definition 7) introduced in Chapter 3 provide efficiently computable upper bounds on $EVC(RS, \mathcal{D})$. Let us note that the s-index crucially hinges on tight and efficient upper bound on the capacity of a sequence. Thus, in the next Section, we provide a novel algorithm to compute an upper bound on the capacity of a sequential pattern.

5.5.1 New Upper Bound on the Capacity of a Sequence

The exact capacity $c(s)$ of a sequence s can be computed using the algorithm described in [16], but it is computationally expensive and may be prohibitive for large datasets. Thus, we are interested in efficiently computable upper bounds on $c(s)$. A first naïve bound, that we denote by $\tilde{c}_n(s) \geq c(s)$, is given by $2^{||s||} - 1$, but it may be a loose upper bound since $c(s) = 2^{||s||} - 1$ if and only if all the items contained in all the itemsets of the sequence s are different.

The second upper bound has been introduced in [75]. Such an upper bound, that we denote by $\tilde{c}(s) \geq c(s)$, can be computed as follows. When s contains, among others, two itemsets A and B s.t. $A \subseteq B$, subsequences of the form $\langle C \rangle$ with $C \subseteq A$ are considered twice in $2^{||s||} - 1$, "generated" once from A and once from B. To avoid over-counting such $2^{|A|} - 1$ subsequences, [75] proposes to consider only the ones "generated" from the longest itemset that can generate them.

In this Section, we introduce a novel, tighter upper bound $\hat{c}(s) \geq c(s)$. Our upper bound is based on the following observation. Let itemsets A and B be respectively the *i-th* and *j-th* itemset of the sequence s with $i < j$, that is, A comes before B in s, and let $T = A \cap B \neq \emptyset$ be their intersection. Let D be a subset of the *bag-union* of the itemsets in s that come before A, that is $D \subseteq \bigcup_{S_k \in s: k<i} S_k$, and let E be a subset of the bag-union of the itemsets in s that come after B, that is $E \subseteq \bigcup_{S_\ell \in s: \ell > j} S_\ell$. The sequences of the form $\langle DCE \rangle$, with $C \subseteq T$, are also considered twice, for the same reasons explained above. Given $a = \sum_{k=1}^{i-1} |S_k|$ the sum of the sizes of the itemsets before A in the sequence s and $b = \sum_{\ell=j+1}^{|s|} |S_\ell|$ the sum of the sizes of the ones that come after B, the number of over-counted sequences of this form is $2^a \cdot (2^{|T|} - 1) \cdot 2^b$. Let us note that this new formula also includes the sequences of the form $\langle C \rangle$, since D and E may be the empty set.

An algorithm to compute an upper bound $\hat{c}(s)$ based on the observation above is given in Algorithm 7. Let $s = \langle S_1, S_2, ..., S_{|S|} \rangle$ be a sequence and assume to *re-label* the itemsets in s by *increasing size*, ties broken arbitrarily, i.e., following the original order. Let $\hat{s} = \langle S_1, S_2, ..., S_{|\hat{s}|}\rangle$ be the sequence in the new order, s.t. $|S_i| \leq |S_{i+1}|, \forall\, i \in \{1, \dots, |\hat{s}| - 1\}$. Let $N = [n_1, n_2, ..., n_{|\hat{s}|}]$ be a vector s.t. its *i-th* element n_i is the sum of the sizes of the itemsets that in the original ordered sequence s come before the *i-th* itemset of the new ordered sequence $\hat{s}$. The inputs of our algorithm are the new ordered sequence $\hat{s}$ and the vector N. First, $\hat{c}(\hat{s})$ is set to $2^{||\hat{s}||} - 1$ (line 2). For each itemset $S_i \in \hat{s}$, we check whether there exists an itemset S_j, with $j > i$, s.t. the set $T_{ij} = S_i \cap S_j$ is non-empty (line 6). For such S_j, we compute the number of over-counted subsequences with the formula above (line 7). In line 7, the *min* and *max* functions are used to check which itemset comes first in the original ordered sequence. After checking the entire sequence $\hat{s}$ for a single itemset S_i, we remove the maximum number of over-counted subsequences found for such S_i (line 9). Then, we update the vector N, subtracting the size of S_i from each n_m, if the itemset m comes after the itemset i in the original ordered sequence s (lines 11-13).

Algorithm 7: `CapUpperBound`: compute the upper bound $\hat{c}(\hat{s})$.

```
Data: Sequence $\hat{s} = \langle S_1, S_2, .., S_{|\hat{s}|}\rangle$, with the $S_i's$ labeled as described in the text,
      vector $N = [n_1, n_2, .., n_{|\hat{s}|}]$, with the $n_i's$ computed as described in the text.
Result: Upper bound $\hat{c}(\hat{s})$ to $c(s)$.
1  $t \leftarrow ||\hat{s}||$;
2  $\hat{c}(\hat{s}) \leftarrow 2^t - 1$;
3  for $i \leftarrow 1$ to $|\hat{s}| - 1$ do
4      $val \leftarrow 0$;
5      for $j \leftarrow i + 1$ to $|\hat{s}|$ do
6          if $\exists\, T = S_i \cap S_j \;:\; T \neq \emptyset$ then
7              $val \leftarrow \max\{val, 2^{\min(n_i, n_j)} \cdot \left(2^{|T|} - 1\right) \cdot$
               $\cdot\, 2^{t - \max(n_i + |S_i|, n_j + |S_j|)}\}$;
8      if $val \neq 0$ then
9          $\hat{c}(\hat{s}) \leftarrow \hat{c}(\hat{s}) - val$;
10         $t \leftarrow t - |S_i|$;
11         for $m \leftarrow i + 1$ to $|\hat{s}|$ do
12             if $n_m > n_i$ then
13                 $n_m \leftarrow n_m - |S_i|$;
14 return $\hat{c}(\hat{s})$;
```

Example 5. *Let us consider the sequence* $s = \langle\{1\}, \{2, 5, 7\}, \{4\}, \{2, 3, 5\}, \{1, 8\}\rangle$. *The inputs of our algorithm are* $\hat{s} = \langle\{1\}, \{4\}, \{1, 8\}, \{2, 5, 7\}, \{2, 3, 5\}\rangle$ *and* $N = [0, 4, 8, 1, 5]$. *The naïve upper bound* $\tilde{c}_n(s)$ *is* $2^{10} - 1 = 1023$. *The upper bound* $\tilde{c}(s)$ *defined in [75] is 1022, since it only removes once the sequence* $\langle\{1\}\rangle$. *The upper bound* $\hat{c}(s)$ *obtained with our algorithm is* 1010, *since we remove the sequence* $\langle\{1\}\rangle$ *but also sequences generated by* *the intersection of* $\{2, 5, 7\}$ *and* $\{2, 3, 5\}$ *combined with other itemsets (e.g., the sequence* $\langle\{2, 5\}, \{1, 8\}\rangle$*).*

Using Algorithm 7 one can compute upper bounds on the capacities of the transactions of $\mathcal{D}$, which can be used to obtain the s-index. Such a bound can be used in Theorem 9 as upper bound on the empirical VC-dimension of sequential patterns, in order to compute a bound on the maximum deviation of the true frequencies of sequential patterns. With such a bound on the maximum deviation, we can use gRosSo to find FPF and FNF approximations of the statistically robust sequential patterns.

5.6 Mining Statistically Robust Itemsets

In this Section, we refresh the task of itemset mining (see Section 2.1.1), as an other concrete realization of the general framework of pattern mining we introduced in Section 5.2.1. Then, we apply the VC-dimension to itemsets and we discuss a VC-dimension based strategy to bound the maximum deviation of the true frequencies of itemsets, which can be used in the SRP mining scenario.

Let us remember, from Section 2.1.1 that an *itemset* X is a non-empty subset of $\mathcal{I}$, i.e., $X \subseteq \mathcal{I}$, $X \neq \emptyset$, where $\mathcal{I} = \{i_1, i_2, ..., i_p\}$ is a finite set of items. We denote by $\mathbb{I}$ the set of all possible itemsets composed by items from $\mathcal{I}$. The *length* $|X|$ of X is the number of items in X and an itemset X is contained in an other itemset Y if and only if $X \subseteq Y$.

As we described in Section 5.5 for the sequential patterns, given a dataset $\mathcal{D}$ for the itemset mining task, that is a finite bag of transactions sampled from $\mathbb{I}$ in accordance with π, we aim to compute the empirical VC-dimension $EVC(RS, \mathcal{D})$ of the range space (see Definition 11) associated with $\mathbb{I}$ on the dataset $\mathcal{D}$ in order to find a probabilistic bound $\mu \in (0, 1)$ on the maximum deviation $\sup_{X\in\mathbb{I}} |t_\pi(X) - f_\mathcal{D}(X)|$. In particular, given $EVC(RS, \mathcal{D})$ and using Theorem 9, it is possible to compute a $\mu \in (0, 1)$ s.t. $\sup_{X\in\mathbb{I}} |t_\pi(X) - f_\mathcal{D}(X)| \leq \mu$. The *d-index* introduced by Riondato and Upfal [66] provides an efficiently computable upper bound on $EVC(RS, \mathcal{D})$.

Definition 12 ([66])**.** *Let $\mathcal{D}$ be a dataset for the itemset mining task. The d-index of $\mathcal{D}$ is the maximum integer d such that $\mathcal{D}$ contains at least d different transactions of length at least d, such that no one of them is a subset of another, i.e., the d transactions form an anti-chain.*

The d-index can be used in Theorem 9 as upper bound on the empirical VC-dimension of itemsets in order to compute a bound on the maximum deviation of the true frequencies of itemsets. With such a bound on the maximum deviation, we can use GRosSo to find FPF and FNF approximations of the statistically robust itemsets.

5.7 Experimental Evaluation

In this Section, we report the results of our experimental evaluation on multiple pseudo-artificial datasets to assess the performance of GRosSo for approximating the statistically robust sequential patterns and itemsets. Then, we execute GRosSo on multiple real datasets to approximate the statistically robust sequential patterns and we analyze the sequential patterns mined. To bound the maximum deviations, as required by GRosSo, we use Theorem 9. The VC-dimension of sequential patterns is bounded using the s-index obtained by using our algorithm (Algorithm 7) to compute the upper bound on the capacity of each sequential transaction, while the VC-dimension of itemsets is bounded using the d-index (Definition 12).

The goals of the evaluation are the following:

- Assess the performance of our algorithm to compute an upper bound on the capacity $c(s)$ of a sequence s, comparing our upper bound with the naïve bound and with the one proposed by [75] (see Section 5.5.1).
- Assess the performance of GRosSo on pseudo-artificial datasets to mine statistically robust sequential patterns and itemsets, checking whether, with probability $1 - \delta$, the set of patterns returned by GRosSo does not contain false positives or false negatives.
- Assess the performance of GRosSo to mine statistically robust sequential patterns on real datasets.

Since this is the first work that considers the problem of mining SRPs, there are not methods to compare with.

5.7.1 Implementation, Environment, and Datasets

We implemented GRosSo for mining statistically robust sequential patterns and itemsets, and our algorithm to compute an upper bound on the capacity of a sequence in Java. To mine the frequent sequential patterns and frequent itemsets, we used, respectively, the PrefixSpan [54] and the FP-Growth [29] implementations both provided by the SPMF library [18]. We performed all experiments on the same machine with 512 GB of RAM and 2 Intel(R) Xeon(R) CPU E5-2698 v3 @ 2.3GHz, running Ubuntu 14.14 and using Java 1.8.0_201. Our open-source implementation and the code developed for the tests and to generate the datasets are available at https://github.com/VandinLab/gRosSo. In all experiments, we fixed $\delta = 0.1$.

Here, we provide the details on the generation of the real datasets for the sequential pattern mining task. The details on the generation of pseudo-artificial datasets for the sequential pattern and itemset mining tasks are provided in the respective Sections. To obtain sequences of real sequential datasets, we generated multiple datasets starting from the Netflix Prize data,[1] which contains over 100 million ratings from 480 thousand randomly-chosen anonymous Netflix customers over 17 thousand movie titles collected between October 1998 and December 2005.

To generate a single dataset, we collected all the movies that have been rated by the users in a given time interval (e.g., in 2004). Each transaction is the temporal ordered sequence of movies rated by a single user, with the movies sorted by ratings' date. Movies rated by such a user in the same day form an itemset and each movie is represented by its year of release. Considering consecutive time intervals, we obtained a sequence of datasets, where each dataset only contains data generated in a single time interval. From the original data we removed movies which year of release is not available and movies that have been rated in a year that is antecedent to their year of release. The latter are due to one of the perturbations introduced in the data to preserve the privacy of the users.[2]

We considered the data collected between January 2003 and December 2005. For each year 2004 and 2005, we generated two types of sequences: the first one composed by 4 datasets, e.g., 2004(Q1-Q4) (each dataset contains the data generated in 3 months), and the second one composed by 3 datasets, e.g., 2004(T1-T3) (each dataset contains the data generated in 4 months). Finally, we generated another sequence of datasets, 2003-2005, considering the entire data between 2003 and 2005 (each dataset contains the data generated in one year).

The characteristics of the generated real datasets are reported in Table 5.1.

5.7.2 Upper Bound on the Capacity

In this Section, we report the results of Algorithm 7, which computes the upper bound $\hat{c}(s)$ on the capacity of a sequence, and compare it with the naïve upper bound $\tilde{c}_n(s) = 2^{||s||} - 1$, and the upper bound $\tilde{c}(s)$ from [75]. (See Section 5.5.1.)

[1] https://www.kaggle.com/netflix-inc/netflix-prize-data
[2] https://en.wikipedia.org/wiki/Netflix_Prize

Table 5.1 Real datasets characteristics and comparison of the upper bounds on the capacity. The Table reports: Dataset $\mathcal{D}$: name of the real dataset; $|\mathcal{D}|$: number of transactions; $|\mathcal{I}|$: total number of items; Avg $||\tau||$: average transaction item-length; $\Delta_{no}(\%)$ and $\Delta_{po}(\%)$: average relative differences between our upper bound on the capacity and the previously proposed ones. The datasets are grouped in sequences.

Dataset $\mathcal{D}$	$\lvert\mathcal{D}\rvert$	$\lvert\mathcal{I}\rvert$	Avg. $\lVert\tau\rVert$	$\Delta_{no}(\%)$	$\Delta_{po}(\%)$
2004Q1	132,907	93	24.2	11.42	10.55
2004Q2	165,428	93	23.5	11.61	10.76
2004Q3	184,109	93	24.7	9.18	8.48
2004Q4	218,151	93	24.9	9.77	9.00
2005Q1	266,799	94	26.2	12.31	11.34
2005Q2	291,627	94	25.3	12.15	11.14
2005Q3	315,316	94	24.7	8.67	7.86
2005Q4	295,797	94	19.9	7.89	6.74
2004T1	152,657	93	29.2	11.64	10.94
2004T2	184,202	93	30.3	11.64	10.96
2004T3	229,929	93	30.6	9.71	9.09
2005T1	290,287	94	32.0	13.03	12.17
2005T2	331,117	94	31.4	11.14	10.38
2005T3	326,668	94	25.7	8.53	7.61
2003Y	117,497	92	51.6	13.81	13.37
2004Y	259,407	93	65.9	11.91	11.54
2005Y	451,435	94	62.2	12.07	11.71

Table 5.1 shows the averages (over all transactions) of the relative differences between our novel upper bound $\hat{c}(s)$ and the previously proposed ones, which, for a dataset $\mathcal{D}$, are computed as

$$\Delta_{no}(\%) = \frac{1}{|\mathcal{D}|}\sum_{\tau\in\mathcal{D}}\left(\frac{\tilde{c}_n(\tau)-\hat{c}(\tau)}{\tilde{c}_n(\tau)}\right)\cdot 100$$

and

$$\Delta_{po}(\%) = \frac{1}{|\mathcal{D}|}\sum_{\tau\in\mathcal{D}}\left(\frac{\tilde{c}(\tau)-\hat{c}(\tau)}{\tilde{c}(\tau)}\right)\cdot 100.$$

In all the datasets, our novel bound is (on average) tighter than the other bounds, with a maximum improvement of 13.81% on the naïve method and 13.37% on the method proposed by [75].

5.7.3 Results with Pseudo-Artificial Datasets

In this Section, we report the results of our evaluation on pseudo-artificial datasets. First, we describe the experimental evaluation using pseudo-artificial datasets for the sequential pattern mining task and then for the itemset mining task.

Sequential Patterns

Here, we report the results of our experimental evaluation using pseudo-artificial datasets for the sequential pattern mining task. We considered the 2005(T1-T3) sequence of datasets as *ground truth* for the sequential patterns, and we generated random datasets taking random samples from each of the datasets in the sequence. In such a way, we know the true frequencies of the sequential patterns (the probability that a pattern belongs to a transaction sampled from a dataset is exactly the frequency that such a pattern has in that dataset). Then, we executed gRosSo on the pseudo-artificial datasets and, by knowing the true frequencies of the patterns, we assessed its performance in terms of false positives, false negatives, and of correctly reported patterns. Since it is not feasible to obtain all the statistically robust sequential patterns due to the gargantuan number of candidates to consider in such datasets, for the EP and DP scenarios, we only considered patterns with true frequency above a minimum threshold θ in the last and first dataset, respectively, while for the SP with true frequency above θ in all the datasets, as defined in Section 5.3.

From each of the three original datasets, 2005T1, 2005T2 and 2005T3, we generated a random dataset with the same size of the corresponding original one, obtaining a sequence of three random datasets. From such a sequence, we mined the set of statistically robust sequential patterns without considering the uncertain of the data, i.e., directly using Equation 5.1, Equation 5.2, or Equation 5.3, using the observed frequencies of the patterns in the random datasets. This allows us to verify whether the set of sequential patterns obtained considering only the frequencies (i.e., without taking the uncertainty into account) results in false positives or in false negatives.

We then ran gRosSo on the sequence of random datasets to mine a FPF or a FNF approximation of the statistically robust sequential patterns, and checked whether the returned approximation contained, respectively, false positives or false negatives. We also reported what fraction of statistically robust sequential patterns is reported by gRosSo. (For both gRosSo and the observed frequency-based approach above, we only considered patterns with frequency greater than θ as explained above, matching our ground truth.)

Table 5.2 Results on pseudo-artificial datasets for EP and DP for the sequential pattern mining task with guarantees on the false positives. The Table reports: $\mathcal{D}_1^n$: name of the sequences of datasets; ε: emerging threshold; θ: minimum frequency threshold; for both the EP and DP: $|GT|$: number of SRPs in the ground truth; T.FP$_f$: percentage of times that the SRPs mined using the observed frequencies contain false positives; T.FP$_g$: percentage of times that the SRPs mined using gRosSo contain false positives; $|\mathcal{A}_P|/|GT|$: average ratio between reported patterns by gRosSo and patterns in the ground truth over 5 random sequences.

Datasets $\mathcal{D}_1^n$	ε	θ	EP				DP			
			$\|GT\|$	T.FP$_f$	T.FP$_g$	$\|\mathcal{A}_P\|/\|GT\|$	$\|GT\|$	T.FP$_f$	T.FP$_g$	$\|\mathcal{A}_P\|/\|GT\|$
$\mathcal{S}_1^n$	0.01	0.3	18	0%	0%	0.46	245	60%	0%	0.28
		0.2	104	0%	0%	0.21	2439	100%	0%	0.08
$\mathcal{S}_1^{n\times 2}$	0.01	0.3	18	0%	0%	0.62	245	60%	0%	0.48
		0.2	104	20%	0%	0.38	2439	100%	0%	0.23
$\mathcal{S}_1^{n\times 3}$	0.01	0.3	18	0%	0%	0.67	245	60%	0%	0.58
		0.2	104	60%	0%	0.43	2439	100%	0%	0.34

Table 5.3 Results on pseudo-artificial datasets for SP for the sequential pattern mining task with guarantees on the false positives. The Table reports: α: stability threshold. See Table 5.2 for the meaning of the other values.

Datasets $\mathcal{D}_1^n$	α	θ	$\|GT\|$	T.FP$_f$	T.FP$_g$	$\|\mathcal{A}_P\|/\|GT\|$
$\mathcal{S}_1^n$	0.1	0.3	42	60%	0%	0.02
		0.2	430	100%	0%	0.06
$\mathcal{S}_1^{n\times 2}$	0.1	0.3	42	40%	0%	0.29
		0.2	430	100%	0%	0.30
$\mathcal{S}_1^{n\times 3}$	0.1	0.3	42	40%	0%	0.49
		0.2	430	100%	0%	0.46

Table 5.2 reports the average results, over 5 different random sequences, denoted by $\mathcal{S}_1^n$, for mining FPF approximations of the EP and DP with $\varepsilon \in \{0, 0.01, 0.05\}$, Table 5.3 reports the average results for mining FPF approximations of the SP with $\alpha \in \{0.05, 0.1\}$, while Table 5.4 reports the average results for mining FNF approximations of the EP. We repeated the entire procedure with 5 sequences of random datasets, denoted by $\mathcal{S}_1^{n\times 2}$, where each random dataset had size twice the original one, and then with five sequences of random datasets, denoted by $\mathcal{S}_1^{n\times 3}$, with size three times the original one. For all the experiments, we used $\theta \in \{0.2, 0.3\}$. Here, we report only a representative subset of the results, with $\varepsilon = 0.01$ and $\alpha = 0.1$. Other results are analogous and discussed below.

Table 5.4 Results on pseudo-artificial datasets for EP for the sequential pattern mining task with guarantees on the false negatives. The Table reports: T.FN$_f$: percentage of times that the SRPs mined using the observed frequencies contain false negatives; T.FN$_g$: percentage of times that the SRPs mined using gRosSo contain false negatives; $|GT|/|\mathcal{A}_N|$: average ratio of patterns in the ground truth and reported patterns by gRosSo over 5 random sequences. See Table 5.2 for the meaning of the other values.

Datasets $\mathcal{D}_1^n$	ε	θ	$\|GT\|$	T.FN$_f$	T.FN$_g$	$\|GT\|/\|\mathcal{A}_N\|$
$\mathcal{S}_1^n$	0.01	0.3	18	60%	0%	0.45
		0.2	104	80%	0%	0.09
$\mathcal{S}_1^{n\times 2}$	0.01	0.3	18	0%	0%	0.68
		0.2	104	20%	0%	0.35
$\mathcal{S}_1^{n\times 3}$	0.01	0.3	18	0%	0%	0.75
		0.2	104	40%	0%	0.50

The results show that, for almost all parameters, the sets of patterns mined in the pseudo-artificial datasets only considering the observed frequency of the patterns (i.e., without considering the uncertainty) contain false positives or false negatives with high probability. In addition, such a probability increases with a lower θ, and thus with a large number of patterns. Instead, the patterns returned by gRosSo do not contain false positives or false negatives in all the runs and with all the parameters. The results are even better than the theoretical guarantees, since theory guarantees us a probability at least $1-\delta = 0.9$ of obtaining a set without false positives or without false negatives. Let us note that in some cases, the percentage of reported SRPs is small, in particular for the SP. However, such a percentage increases with larger datasets, since techniques from statistical learning theory, such as the VC-dimension, perform better when larger collections of data are available. In the EP scenario, for both types of approximations, FPF and FNF, we also checked whether the approximations returned by gRosSo had the additional guarantees described in Section 5.4.5, and, in all the runs, we found that such additional guarantees were always respected.

For the EP and DP with guarantees on the false positives, the results obtained with $\varepsilon = 0$ are very close to the ones reported by Table 5.2, in many cases even better, while with $\varepsilon = 0.05$ gRosSo reported a lower percentage (between 0.003 and 0.23) of statistically robust sequential patterns, in particular for the DP scenario. For the SP instead, using $\alpha = 0.05$, we found only few real SRPs in the original data, and gRosSo did not report any of them, while for the EP with guarantees on the false negatives, the results obtained with $\varepsilon = 0$ and $\varepsilon = 0.05$ are very close to the ones reported by Table 5.4, with a percentage of reported patterns between 0.06 and 0.78.

For the EP and DP scenarios with guarantees on the false positives, we also performed an additional experiment to verify the absence of false positives in the output of gRosSo. We generated a random sequence of datasets taking three random samples from the same original dataset 2005T1. In such a way, the random sequence did not contain any EP and DP, since each pattern had the same true frequency in all the datasets. Then, we executed gRosSo on such a sequence using $\theta = 0$ and $\varepsilon = 0$. Let us note that this choice of parameters is the most challenging scenario, since we searched for all the EP and DP we were able to find. Again, we repeated such an experiment with five different random sequences where each dataset had the same size of the original one, five sequences with double size and five sequences with datasets that had three times the size of the original one. In all the runs, gRosSo correctly did not report any EP and DP.

These results show that, in general, considering the observed frequencies of the patterns is not enough to find sets of SRPs that do not contain false positives or false negatives. Thus, techniques like the one introduced in this Chapter are necessary to find large sets of SRPs without false positives or false negatives. In addition, gRosSo is an effective tool to find rigorous approximations of the statistically robust sequential patterns.

Itemsets

Here, we report the results of our experimental evaluation using pseudo-artificial datasets for the itemset mining task. Starting from the 2005(T1-T3) sequence of datasets for the sequential pattern mining task, we first generated the corresponding sequence of datasets, 2005(T1-T3)IT, for the itemset mining task. For each dataset in the sequence 2005(T1-T3), we generated a new dataset taking the union of the items in each transaction of the dataset, e.g., a sequential transaction $\tau = \langle\{1\}, \{2\}, \{6, 7\}, \{2\}\rangle$ becomes a transaction $\tau' = \{1, 2, 6, 7\}$. Then, we considered the 2005(T1-T3)IT sequence as ground truth for the itemsets, and we performed the same experimental evaluation described in the previous Section for the sequential patterns. We denote by $\mathcal{Q}_1^n$, $\mathcal{Q}_1^{n\times 2}$, and $\mathcal{Q}_1^{n\times 3}$ the analogous to $\mathcal{S}_1^n$, $\mathcal{S}_1^{n\times 2}$, and $\mathcal{S}_1^{n\times 3}$, but for itemsets. Table 5.5 reports the average results for mining FPF approximations of the EP and DP, Table 5.6 reports the average results for mining FPF approximations of the SP, while Table 5.7 reports the average results for mining FNF approximations of the EP. The results show that, for almost all parameters, the sets of patterns mined in the pseudo-artificial datasets only considering the observed frequency of the patterns contain false positives or false negatives with high probability. Instead, the

Table 5.5 Results on pseudo-artificial datasets for EP and DP for the itemset mining task with guarantees on the false positives. See Table 5.2 for the meaning of the values.

Datasets $\mathcal{D}_1^n$	ε	θ	EP				DP			
			$\|GT\|$	T.FP$_f$	T.FP$_g$	$\|\mathcal{A}_P\|/\|GT\|$	$\|GT\|$	T.FP$_f$	T.FP$_g$	$\|\mathcal{A}_P\|/\|GT\|$
$\mathcal{Q}_1^n$	0.01	0.3	26	20%	0%	0.17	6	0%	0%	0.33
		0.2	48	60%	0%	0.14	60	80%	0%	0.04
$\mathcal{Q}_1^{n\times 2}$	0.01	0.3	26	0%	0%	0.36	6	100%	0%	0.67
		0.2	48	100%	0%	0.20	60	100%	0%	0.07
$\mathcal{Q}_1^{n\times 3}$	0.01	0.3	26	100%	0%	0.42	6	100%	0%	0.67
		0.2	48	100%	0%	0.23	60	100%	0%	0.07

Table 5.6 Results on pseudo-artificial datasets for SP for the itemset mining task with guarantees on the false positives. See Table 5.3 for the meaning of the values.

Datasets $\mathcal{D}_1^n$	α	θ	$\|GT\|$	T.FP$_f$	T.FP$_g$	$\|\mathcal{A}_P\|/\|GT\|$
$\mathcal{Q}_1^n$	0.1	0.3	419	80%	0%	0.65
		0.2	10541	60%	0%	0.70
$\mathcal{Q}_1^{n\times 2}$	0.1	0.3	419	0%	0%	0.81
		0.2	10541	0%	0%	0.76
$\mathcal{Q}_1^{n\times 3}$	0.1	0.3	419	0%	0%	0.82
		0.2	10541	0%	0%	0.78

Table 5.7 Results on pseudo-artificial datasets for EP for the itemset mining task with guarantees on the false negatives. See Table 5.4 for the meaning of the values.

Datasets $\mathcal{D}_1^n$	ε	θ	$\|GT\|$	T.FN$_f$	T.FN$_g$	$\|GT\|/\|\mathcal{A}_N\|$
$\mathcal{Q}_1^n$	0.01	0.3	26	80%	0%	0.20
		0.2	48	60%	0%	0.02
$\mathcal{Q}_1^{n\times 2}$	0.01	0.3	26	0%	0%	0.24
		0.2	48	0%	0%	0.04
$\mathcal{Q}_1^{n\times 3}$	0.01	0.3	26	0%	0%	0.30
		0.2	48	0%	0%	0.05

patterns returned by gRosSo do not contain false positives or false negatives in all the runs, as observed for the sequential patterns. In addition, all the approximations of the EP reported by gRosSo respected the additional guarantees introduced in Section 5.4.5. All these results emphasize that considering the observed frequency is not enough to find large sets of SRPs without false positives or false negatives, and that gRosSo is an effective tool also to find rigorous approximations of the statistically robust itemsets. Comparing these results with the ones obtained for the sequential patterns, it is interesting to notice that in the EP and DP scenarios, almost always the returned statistically robust itemsets are less than the corresponding statistically robust sequential patterns, in particular for the DP, and that also the percentages of reported patterns are lower for the itemsets. Instead, in the SP scenario, more statistically robust itemsets are returned, always with higher percentages of reported patterns.

5.7.4 Results with Real Datasets

Here, we report the results of gRosSo for mining statistically robust sequential patterns from the Netflix real datasets. First, we report and discuss the results with guarantees on the false positives for all the three types of SRPs. For the EP and DP, we did not use any constraints on the minimum frequency, thus we reported every statistically robust sequential patterns found in the data. Table 5.8 shows the results for the EP and DP with guarantees on the false positives. In the EP scenario, for the sequences of datasets composed by four datasets (denoted by Q1-Q4), gRosSo reported only few patterns. In particular, all the emerging sequential patterns returned contain the year of the dataset in which they were found, e.g., in 2004(Q1-Q4) all the EP contain the item 2004, with a frequency close to zero in the first dataset. Since during the year many more movies come out, the number of users that rates such movies increases through the year and so such patterns emerge through the sequence. We found the same result in sequences composed by three datasets (denoted by T1-T3) but in this case gRosSo reported many more patterns, in particular for the 2004 sequence, since now we were considering the emerging condition only in three datasets, and thus patterns with such an emerging behavior are easier to discover.

gRosSo did not report any DP in all the datasets using $\varepsilon = 0.05$. Observing the patterns found on 2005(T1-T3), we noted that the maximum absolute difference $\max_{s \in \mathcal{A}} |f_{\mathcal{D}_1}(s) - f_{\mathcal{D}_n}(s)|$ over all the returned patterns between the frequency of a pattern in the first dataset and its frequency in the last dataset was 0.26, while for the EP such a difference was 0.60. Thus, while the frequencies of the EP increase a lot through the year, the frequencies of the DP decrease less, which explains why fewer descending patterns

Table 5.8 Results on real datasets for EP and DP for the sequential pattern mining task with guarantees on the false positives. The Table reports: $\mathcal{D}_1^n$: name of the sequences of datasets; ε: emerging threshold; for both the EP and DP: $|\mathcal{A}_P|$: number of returned SRPs; Avg$||s||$: average item-length of the returned SRPs.

Datasets $\mathcal{D}_1^n$	ε	EP		DP	
		$\lvert\mathcal{A}_P\rvert$	Avg$\lvert\lvert s\rvert\rvert$	$\lvert\mathcal{A}_P\rvert$	Avg$\lvert\lvert s\rvert\rvert$
2004(Q1-Q4)	0	25	2.4	0	/
	0.01	16	2.3	0	/
	0.05	1	1.0	0	/
2005(Q1-Q4)	0	2	1.5	10	3.2
	0.01	1	1.0	2	2.5
	0.05	0	/	0	/
2004(T1-T3)	0	5213	4.6	5	3.4
	0.01	2214	4.4	0	/
	0.05	207	3.6	0	/
2005(T1-T3)	0	113	3.6	689	5.4
	0.01	48	3.3	187	4.9
	0.05	4	2.5	0	/
2003-2005(Y)	0.05	15107	5.4	14	5.5

Table 5.9 Results on real datasets for SP for the sequential pattern mining task with guarantees on the false positives. The Table reports: α : stability threshold; θ : minimum frequency threshold. See Table 5.8 for the meaning of the other values.

Datasets $\mathcal{D}_1^n$	α	θ	$\lvert\mathcal{A}_P\rvert$	Avg$\lvert\lvert s\rvert\rvert$
2004(Q1-Q4)	0.1	0.4	2	1.0
		0.2	40	1.8
2005(Q1-Q4)	0.1	0.4	0	/
		0.2	7	1.7
2004(T1-T3)	0.1	0.4	3	1.0
		0.2	146	2.2
2005(T1-T3)	0.1	0.4	1	1.0
		0.2	18	2.1
2003-2005(Y)	0.1	0.4	3	2.0
		0.2	458	3.9

Table 5.10 Results on real datasets for EP for the sequential pattern mining task with guarantees on the false positives and on the false negatives. The Table reports: $\mathcal{D}_1^n$: name of the sequences of datasets; θ : minimum frequency threshold; ε: emerging threshold; for both the FPF and FNF approximations: $|\mathcal{A}|$: number of returned SRPs; Avg$||s||$: average item-length of the returned SRPs.

Datasets $\mathcal{D}_1^n$	θ	ε	FPF		FNF	
			$\|\mathcal{A}_P\|$	Avg$\|\|s\|\|$	$\|\mathcal{A}_N\|$	Avg$\|\|s\|\|$
2004(Q1-Q4)	0.3	0	21	2.1	119	2.3
		0.01	15	2.1	114	2.3
		0.05	1	1.0	80	2.6
2005(Q1-Q4)	0.3	0	2	1.5	13	2.0
		0.01	1	1.0	11	2.1
		0.05	0	/	4	1.8
2004(T1-T3)	0.3	0	74	2.7	309	2.6
		0.01	73	2.7	298	2.6
		0.05	63	2.7	243	2.8
2005(T1-T3)	0.3	0	9	2.0	43	2.0
		0.01	8	2.3	41	2.0
		0.05	3	2.2	25	2.0
2003-2005(Y)	0.5	0	46	2.2	255	2.1
		0.01	44	2.1	242	2.1
		0.05	41	2.1	227	2.1

are found by gRosSo. The DP found on 2005(T1-T3) are on average larger than the EP found on the same data, and the 96% of such patterns contain the item 2004, many of them multiple times. Thus, they probably represent long sequential patterns whose frequencies decrease, since the users watch always less 2004's movies through the year 2005 and so, it is difficult for such long patterns to persist through the time.

Table 5.9 shows the results for the SP with guarantees on the false positives. We performed experiments varying $\theta \in \{0.2, 0.4\}$ and $\alpha \in \{0.05, 0.1\}$. With $\alpha = 0.05$, gRosSo did not report any SP for all the datasets. Almost always the SP found by gRosSo are quite short combinations of items that represents movies of the 90s or early 2000s, that precede the year of the mined sequence. It is surprising that sequential patterns that contain such "old" items are stable through the time, e.g., $\langle\{2000, 2001\}, \{1990\}\rangle$ has a maximum absolute difference between all its frequencies of 0.025 in the sequence 2003-2005(Y). Such sequential patterns probability represent some classical movies that people always watch with the same frequency through time.

To conclude, we report the results for the EP with guarantees on the false negatives. As we discussed in Section 5.4.4, we decided to consider only patterns p with $t_{\pi_n}(p) \geq \theta$ to reduce the amount of starting candidates. In order to compare the size of FPF and FNF approximations in the same scenario, we also executed the same experiments for the EP with guarantees on the false positives using a minimum frequency threshold. Table 5.10 reports the parameters and the results of such experiments. These results show that gRosSo can detect EP when one is interested in finding FPF or FNF approximations, and it is possible to notice that almost always the FNF approximations contain a number of EP that is from 4 to 6 times larger than the corresponding number in the FPF approximations.

Overall, the results show that gRosSo detects various types of SRPs from real datasets, obtaining insights into the evolution of the generative process underlying the data.

Chapter 6

Permutation Strategies for Mining Statistically Significant Sequential Patterns

In this Chapter, we consider the problem of mining statistically significant sequential patterns. Let us remember that in Chapter 4 and 5, we considered the scenario in which the dataset, or the sequence of datasets, are random sample from unknown probability distributions and we studied the problem of mining patterns that are frequency generated by such distributions, or whose true frequencies follow well specified trends though the sequence of datasets. In significant pattern mining, the dataset is still considered a sample from an unknown probability distribution, but the goal is to mine patterns significantly deviating from an assumed null hypothesis. Key challenges is such a scenario are the definition of appropriate null models for the data, the assessment of the significance of the patterns of interest, and the multiple hypothesis testing problem, since one is interested in testing multiple patterns simultaneously. To solve these difficult challenges, we employ the statistical hypothesis testing framework and techniques based on permutation testing. However, let us note that the significant pattern mining task and the ones described in the two previous Chapters are not unrelated. Indeed, an approach that we define in this Chapter to mine significant sequential patterns from massive datasets, is based on techniques developed to solve the tasks of the previous Chapters. Part of the contributions described in this Chapter appear in [83].

6.1 Introduction

While the frequency of a sequential pattern is an important feature in some applications, it is usually not sufficient to identify interesting sequential patterns, i.e., sequential patterns that provide useful knowledge regarding the process described by the data. For example, a sequence of itemsets may appear frequently in a dataset simply because each of the itemsets has high individual frequency, even if there is no association between the itemsets in the sequence. Since sequential patterns are ordered sequences of events, represented by itemsets, in this work we are interested in understanding whether the frequency of a sequential pattern is due to the observed order, i.e., on the relation, between its itemsets, or if it is due to other factors. A natural framework to identify interesting sequential patterns is provided by *statistical hypothesis testing*, where the goal is to mine *statistically significant sequential patterns*, defined as sequences that appear *more frequently than expected* under an appropriate (generative) *null model* for the data.

The extraction of statistically significant patterns has received a lot of attention when patterns are itemsets, with several methods [35, 90, 21, 25] that have been proposed to identify significant itemsets while providing guarantees on the false discoveries (i.e., itemsets flagged as significant while they are not). The most commonly used guarantee is given by the Family-Wise Error Rate (FWER), that is the probability that one or more false discoveries are reported in output. Strikingly, only few methods [24, 47] to identify statistically significant *sequential* patterns have been proposed. The extraction of statistically significant sequential patterns is more complex than the extraction of significant itemsets, mostly for two key issues: first, the number of sequential patterns that can be built from a ground set of items is much larger than the number of itemsets, and the gargantuan number of candidate sequential patterns poses a *multiple hypothesis testing* issue, since the probability of a false discovery increases with the number of candidate patterns; second, the definition of appropriate null models for sequential patterns is more difficult, since reasonable null models do not result in distributions that can be analytically described and it is therefore crucial to be able to efficiently compute the statistical significance of patterns. To the best of our knowledge, no method to rigorously identify statistically significant sequential patterns with rigorous guarantees on the FWER for the reported patterns is available.

6.1.1 Our Contributions

In this Chapter, we focus on the problem of *mining statistically significant sequential patterns* from a transactional dataset. In this regards, our contributions are:

- We introduce a new algorithm, ProMiSe, to identify statistically significant sequential patterns using permutation testing. ProMiSe is the first algorithm to provide rigorous guarantees on the Family-Wise Error Rate (FWER) of the output, using the Westfall-Young method to properly correct for multiple hypothesis testing.
- We introduce and formalize three strategies, based on swaps or on permutations at the level of itemsets, to generate (random) permuted datasets for sequential pattern mining. These three strategies are at the core of ProMiSe, and they sample datasets from the distribution of all datasets where the number of appearances of each itemset and the number of itemsets in each transaction are the same as in the input dataset.
- We provide a formal analysis of the itemsets swapping strategy, proving that a polynomial number of swaps are sufficient to uniformly sample a random dataset from the aforementioned distribution. Moreover, we experimentally show that a number of swaps proportional to the number of itemsets in the dataset is sufficient to sample a random dataset.
- We introduce an alternative version of ProMiSe, I-ProMiSe, that results in a lower statistical power than the original version but that is several orders of magnitude faster, allowing to mine significant sequential patterns from massive datasets.
- We provide an implementation of our algorithms and conduct an extensive experimental evaluation of their usage to extract significant sequential patterns, showing that they allows to efficiently extract significant sequential patterns from real sequential datasets.

6.1.2 Related Work

Several works have been proposed to identify statistically significant itemsets where the significance is defined in terms of the comparison of itemsets statistics (e.g., frequencies or number) with a null mode (e.g., [35, 21]). The one that is most related to ours is the work of Gionis et al. [21], which introduces a swap randomization approach to assess the significance of patterns (e.g., itemsets) in 0-1 datasets. Such a technique cannot be applied to sequential transactions, due to sequential dimension that is absent in 0-1 datasets. In addition, [21] provides only an experimental assessment of the number of swaps required to sample a random dataset, while we also prove a theoretical upper bound.

Few methods [24, 47] have been proposed to mine statistically significant sequential patterns. Gwadera and Crestani [24] proposes a method based on a null model obtained by combining two models at different levels, in particular itemset-wise and sequence-wise, with a maximum entropy model for the itemset-wise level and a mixture model for the sequence-wise level. Our null model is much simpler and allows for the efficient mining of significant sequential patterns, that is instead not possible with the method of [24], in particular for large patterns. More recently, Low-Kam et al. [47] introduce an approach based on the *independence model*, where each itemset appears in a transaction with probability equal to its frequency in the real dataset and independently of all other events. We consider a null model where transactions' lengths are preserved as well, which is more appropriate in cases where different groups of transactions are in the dataset (similarly to what happens for itemsets [21]). In addition, [47] performs a Bonferroni correction assuming that the candidates sequential patterns are only the frequent patterns *observed* in the dataset, while all potential candidate patterns should be considered, as it has been argued for other pattern mining problems [35].

A different line of works identifies significant patterns, including sequential patterns, where the significance is given by the association of the presence of the pattern with a binary label available from each transaction [80, 51, 45, 60]. These methods cannot be applied to identify patterns whose frequency significantly deviates from a null model. Several methods (e.g., [10, 78, 48]) have been proposed to identify *interesting patterns* using some alternative interestingness measure. These measures, and the methods that employ them, are orthogonal to our approach, which focuses on the statistical significance of patterns.

We remind interested readers to the related work of Chapter 3 (Section 3.1.2) for works regarding frequent sequential pattern mining.

6.1.3 Organization of the Chapter

The rest of the Chapter is structured as follows. Section 6.2 contains the definitions and concepts used throughout this Chapter. ProMiSe, our algorithm for mining significant sequential patterns, and the three strategies to efficiently generate random datasets are described in Section 6.3. Section 6.4 introduce I-ProMiSe, an alternative version of our algorithm which is several orders of magnitude faster than the original one, allowing to analyze massive datasets. Section 6.5 reports the results of an extensive suite of experiments performed to evaluate the effectiveness of ProMiSe on synthetic and real datasets.

6.2 Preliminaries

We now provide the definitions and concepts used throughout this Chapter. Section 6.2.1 defines the problem of mining statistically significant sequential patterns, refreshing the notion of sequential pattern mining. Finally, in Section 6.2.2, we introduce the multiple hypothesis testing framework for the significant sequential pattern mining task.

6.2.1 Significant Sequential Pattern Mining

The task of *mining significant sequential patterns* requires to identify sequential patterns whose frequency is *significant*, that is, whose frequency is not due to random fluctuations in the data. Let us remember, from Section 2.1.2, that a *sequential pattern*, or *sequence*, $s = \langle S_1, S_2, \ldots, S_k \rangle$ is a finite ordered sequence of *itemsets* S_i, with $i \in \{1, \ldots, k\}$, which are sets of elements called *items*. The *length* $|s|$ of s is the number of itemsets in s and the *item-length* $||s||$ of s is the sum of the sizes of the itemsets in s. Given a *domain* $\mathbb{S}$, a *dataset* $\mathcal{D}$, which is a bag of *transactions*, i.e., sequential patterns, from $\mathbb{S}$, the task of *frequent sequential pattern (FSP) mining* requires to report all the sequential patterns with frequency $f_{\mathcal{D}}(s)$ in $\mathcal{D}$ at least $\theta \in (0, 1]$, where the frequency $f_{\mathcal{D}}(s)$ is the fraction of transactions of $\mathcal{D}$ to which s belongs. In particular, sequential patterns describe sequences of events or actions that are useful for predictions in many scenarios. Thus, in this Chapter, we are interested in understanding whether the frequency $f_{\mathcal{D}}(s)$ of a sequential pattern s in a dataset $\mathcal{D}$ is due to the observed order, and thus on the relation, between its itemsets, or if it is due to other factors. Two factors that are natural to consider are the number of times the itemsets appears in the dataset and the number of itemsets in the transactions of the dataset (i.e., the length of the transactions). By mining significant sequential patterns, we are then interested in mining sequential patterns whose high frequency in the dataset is due to the order of their itemsets.

To assess the significance of a sequential pattern, the framework of *statistical hypothesis testing* (see Section 2.5.1) is usually employed. For each sequential pattern s, let H_s be the *null hypothesis* that the frequency $f_{\mathcal{D}}(s)$ of s in $\mathcal{D}$ well conforms to its frequency in a *random dataset*, i.e., a dataset taken uniformly at random among all datasets with properties similar to $\mathcal{D}$. As already said, the properties we consider are:

- the number of times each itemset appears in $\mathcal{D}$;
- the length of each transaction in $\mathcal{D}$.

Therefore, under the null hypothesis, $\mathcal{D}$ is a dataset taken uniformly at random from all datasets where each itemset appears the same number of times it appears in $\mathcal{D}$ *and* each transaction has the same length that it has in $\mathcal{D}$. As an example for why is important to preserve these properties in a random dataset, let us consider a sequential dataset from an e-commerce website where each transaction is the ordered list of purchases made by a single user and the itemsets represent products bought together. In such a dataset, the idea is to preserve the products that the users bought together (the itemsets and the number of times they appear) and also to preserve the number of orders made by the users (the length of the transactions).

Example 6. *Let us consider the dataset $\mathcal{D} = \{\tau_1, \tau_2, \tau_3, \tau_4\}$ in Section 2.1.2 (Example 2). A random dataset $\tilde{\mathcal{D}}$ with the same properties of $\mathcal{D}$ is the following:*

$$\begin{aligned}
\tau_1 &= \langle\{3\}, \{2, 6, 7\}\rangle \\
\tau_2 &= \langle\{7\}, \{2\}, \{6, 7\}, \{1, 2, 5, 6\}\rangle \\
\tau_3 &= \langle\{6, 7\}, \{2\}, \{2\}, \{2\}\rangle \\
\tau_4 &= \langle\{1\}, \{2\}, \{1, 4\}, \{2\}\rangle.
\end{aligned}$$

Let us note that each itemset that appears a certain number of times in $\mathcal{D}$ also appears the same number of times in $\tilde{\mathcal{D}}$, and that each transaction has the same length in the two datasets. The item-lengths of the transactions are instead not mandatorily the same: τ_1 has item-length 4 in both datasets but the item-length of τ_2 changes from 5 to 8. Also the frequency of the items (and so of the itemsets and of the sequential patterns) can change: $f_{\mathcal{D}}(\{2\}) = f_{\tilde{\mathcal{D}}}(\{2\}) = 1$ but $f_{\mathcal{D}}(\{6\}) = 1 \neq 3/4 = f_{\tilde{\mathcal{D}}}(\{6\})$.

Let us remember that under the null hypothesis, the frequency of a sequential pattern s is described by a r.v. X_s and in order to assess its significance, a *p-value* p_s is commonly computed. Thus, in our scenario, the p-value p_s of a sequential pattern s is the probability of observing a frequency at least as large as the frequency $f_{\mathcal{D}}(s)$ of s in $\mathcal{D}$ under the null hypothesis, that is,

$$p_s = \Pr[X_s \geq f_{\mathcal{D}}(s) | H_s].$$

Let us note that the statistical hypothesis testing framework requires to be able to compute the p-values p_s but for complex null hypotheses, such as ours, the p-values cannot be computed analytically. However, when one can sample datasets uniformly at random from the distribution described by the null hypothesis, the p-values can be estimated by a *Monte Carlo (MC) procedure* generating M random datasets $\tilde{\mathcal{D}}_i$ as described in Section 2.5.1, that is,

$$p_s = \frac{1}{M+1}\left(1 + \sum_{i=1}^{M} \mathbb{1}[f_{\tilde{\mathcal{D}}_i}(s) \geq f_{\mathcal{D}}(s)]\right). \tag{6.1}$$

6.2.2 Multiple Hypothesis Testing

Let us remember from Section 2.5.1 that the statistical hypothesis testing framework is commonly used to provide guarantees on the *false discoveries*. When a single sequential pattern s is tested for significance, flagging it as significant when $p_s \leq \alpha$, where $\alpha \in (0,1)$ is a threshold fixed by the user, guarantees that the probability that s corresponds to a false discovery is at most α.

The situation is completely different when several sequential patterns are tested simultaneously since the expected number of false discoveries, for a fixed α, linearly grows with the size of the set of tested hypotheses, posing a severe *multiple hypothesis correction problem* [58]. To solve this issue, a common solution is to identify a *corrected* threshold $\delta \in (0,1)$ s.t. all patterns with p-value $\leq \delta$ can be reported as significant while providing guarantees on the probability of reporting at least one false positive, i.e., the *Family-Wise Error Rate (FWER)*.

One approach to set δ is to use the *Bonferroni correction* [11], setting $\delta = \alpha/d$. However, when d is large, δ is very close to 0, resulting in low *statistical power* with many *false negatives* (i.e., significant patterns that are not reported in output). Let us note that this issue is particular severe for sequential patterns, since if one does not restrict (before analyzing the dataset) the space of sequential patterns, i.e., hypotheses, the number of candidate patterns is *infinite*, and therefore $d = \infty$. Even restricting the set of patterns, for example considering only sequential patterns of item-length at most ℓ, may result in low statistical power, since the number of candidate patterns increases exponentially with ℓ.

More sophisticated techniques have been designed to increase the statistical power, such as the *Westfall-Young (WY) method* [95]. As we discussed in Section 2.5.1, it directly estimates the joint distribution of null hypotheses using P datasets obtained from the distribution described by the null hypothesis, i.e., P random datasets. For every random dataset $\tilde{\mathcal{D}}_i$, with $i \in \{1, \ldots, P\}$, it computes the minimum p-value $p_{min}^{(i)}$ over all sequential patterns of interest in $\tilde{\mathcal{D}}_i$. Finally, it estimates the corrected significance threshold δ^* as the α-quantile of the P minimum p-values, that is,

$$\delta^* = \max\left\{\delta : \sum_{i=1}^{P} \mathbb{1}\left[p_{min}^{(i)} \leq \delta\right] \leq \alpha P\right\}, \tag{6.2}$$

with $\alpha \in (0,1)$ the *FWER threshold*.

6.3 ProMiSe: Mining Significant Sequential Patterns with Permutation Testing

In this Section, we describe ProMiSe, PeRmutatiOn strategies for MIning statistically significant SEquential patterns, our algorithm to mine significant sequential patterns while controlling the probability of reporting at least one false positive, i.e., the FWER. ProMiSe crucially hinges on the ability of sampling random datasets uniformly at random from the distribution described by the null hypothesis, then, in Section 6.3.1, we illustrate three strategies to efficiently generate random datasets. Finally, in Section 6.3.2, we provide the details of our parallel implementation of ProMiSe.

Algorithm 8: ProMiSe

```
Data: Sequential Dataset $\mathcal{D}$, Minimum Frequency Threshold $\theta \in (0,1]$, FWER
      Threshold $\alpha \in (0,1)$.
Result: Set $\mathcal{SSP}$ of significant sequential patterns with FWER $\leq \alpha$.
$\mathcal{F} \leftarrow FSP(\mathcal{D}, \theta)$;
for $i \leftarrow 1$ to $M$ do
    $\tilde{\mathcal{D}}_i \leftarrow$ RandomDataset($\mathcal{D}$);
foreach $s \in \mathcal{F}$ do
    $p_s \leftarrow \frac{1}{M+1}(1 + \sum_{i=1}^{M} \mathbb{1}[f_{\tilde{\mathcal{D}}_i}(s) \geq f_{\mathcal{D}}(s)])$;
foreach $j \leftarrow 1$ to $P$ do
    $\tilde{\mathcal{D}}^{(j)} \leftarrow$ RandomDataset($\mathcal{D}$);
    $\mathcal{F}^{(j)} \leftarrow FSP(\tilde{\mathcal{D}}^{(j)}, \theta)$;
    if $\mathcal{F}^{(j)} = \emptyset$ then $p_{min}^{(j)} \leftarrow \alpha$;
    else
        for $i \leftarrow 1$ to $M$ do
            $\tilde{\mathcal{D}}_i^{(j)} \leftarrow$ RandomDataset($\tilde{\mathcal{D}}^{(j)}$);
        for $\tilde{s} \in \mathcal{F}^{(j)}$ do
            $p_{\tilde{s}} \leftarrow \frac{1}{M+1}(1 + \sum_{i=1}^{M} \mathbb{1}[f_{\tilde{\mathcal{D}}_i^{(j)}}(\tilde{s}) \geq f_{\tilde{\mathcal{D}}^{(j)}}(\tilde{s})])$;
        $p_{min}^{(j)} \leftarrow \min\{p_{\tilde{s}} : \tilde{s} \in \mathcal{F}^{(j)}\}$;
$\delta^* \leftarrow \min\left\{\alpha, \max\left\{\delta : \sum_{j=1}^{P}\left(\mathbb{1}[p_{\min}^{(j)} \leq \delta]\right) \leq \alpha P\right\}\right\}$;
$\mathcal{SSP} \leftarrow \{(s, f_{\mathcal{D}}(s), p_s) : s \in \mathcal{F} \wedge p_s < \delta^*\}$;
return $\mathcal{SSP}$;
```

The pseudo-code of ProMiSe is described in Algorithm 8. Given a sequential dataset $\mathcal{D}$, a minimum frequency threshold $\theta \in (0,1]$, and a FWER threshold $\alpha \in (0,1)$ as input, ProMiSe identifies a set of significant sequential patterns with frequency at least θ and FWER bounded by α. Let us note that in Algorithm 8, we consider the values M

and P, which are, respectively, the number of random datasets to generate for the MC procedure and for the WY method, as parameters fixed by the users. (A discussion about appropriate values for such parameters is provided in Section 6.5.) ProMiSe starts by mining the set $FSP(\mathcal{D}, \theta)$ of sequential patterns with frequency at least θ in $\mathcal{D}$ (line 1). The extraction of such sequential patterns can be performed with any efficient algorithm for the mining of frequent sequential patterns. Then, it uses the MC procedure to estimate the p-value p_s for each sequential pattern $s \in FSP(\mathcal{D}, \theta)$ using Equation 6.1, and thus considering M random datasets (lines 2-5). The procedure `RandomDataset(`$\mathcal{D}$`)` is used to sample a dataset uniformly at random among all datasets where each itemset appears the same number of times it appears in $\mathcal{D}$ and each transaction has the same length that has in $\mathcal{D}$. How to implement such a procedure is illustrated in the next Section. In particular, to generate random datasets, Algorithm 9, 10 or 11 can be employed. Then, ProMiSe employs the WY method to compute the corrected significance threshold δ^* considering P random datasets. For each random dataset $\tilde{\mathcal{D}}^{(j)}$, with $j \in \{1, \dots, P\}$, generated with the same procedure `RandomDataset(`$\mathcal{D}$`)` described above (line 7), it mines the set $FSP(\tilde{\mathcal{D}}^{(j)}, \theta)$ (line 8) and computes the minimum p-value $p_{min}^{(j)}$ over all the sequential patterns $\tilde{s} \in FSP(\tilde{\mathcal{D}}^{(j)}, \theta)$ (line 15). To compute such p-values, it employs a MC procedure analogous of the one described above (lines 11-14). Let us note that if the set $FSP(\tilde{\mathcal{D}}^{(j)}, \theta) = \emptyset$, then we consider $p_{min}^{(j)} = \alpha$, corresponding to an uncorrected threshold (line 9). Finally, it estimates the corrected significance threshold δ^* using Equation 6.2 as the α-quantile of the P minimum p-values $p_{min}^{(j)}$, with $j \in [1, P]$. If $\delta^* > \alpha$, then we set $\delta^* = \alpha$, corresponding to an uncorrected threshold. Finally, the output is the set of sequential patterns $s \in FSP(\mathcal{D}, \theta)$ s.t. $p_s < \delta^*$ (line 17).

Let us note that since we are only interested in sequential patterns with frequency $\geq \theta$, the computation of the minimum p-values $p_{\text{min}}^{(1)}, \dots, p_{\text{min}}^{(P)}$ for the WY method is restricted to sequential patterns that have frequency $\geq \theta$ in the random datasets. That is, $p_{\text{min}}^{(j)}$ is computed as the minimum p-value over all the sequential patterns in $FSP(\tilde{\mathcal{D}}^{(j)}, \theta)$. This corresponds to employ the WY method to estimate the probability of observing *any* sequential pattern with frequency $\geq \theta$ and with p-value below the threshold δ under the null hypothesis, that is what we need in order to estimate the FWER FWER(δ) obtained using δ as significance threshold when we are interested only in sequential patterns with frequency $\geq \theta$.[1] Again, let us note that any efficient implementation for mining frequent sequential patterns can be used to obtain $FSP(\tilde{\mathcal{D}}^{(j)}, \theta)$.

[1]Let us note that in this way we are still allowing patterns s with frequency $f_{\mathcal{D}}(s) < \theta$ to appear with frequency higher than θ in a random dataset, thus properly accounting for such sequential patterns in our multiple hypothesis correction.

The following result, easily derived by the properties of the WY method, establishes the quality of the output of ProMiSe.

Theorem 18. *The output of* ProMiSe *has FWER* $\leq \alpha$.

Proof. Let us consider the P random datasets $\tilde{\mathcal{D}}^{(j)}$, with $j \in \{1, \ldots, P\}$, generated by ProMiSe for the WY method. Let us note that they do not contain any significant sequential patterns by construction, since they are sampled uniformly at random from the distribution described by the null hypothesis. Given $\delta \in (0, 1)$, the FWER FWER(δ) obtained using δ as significance threshold can be estimated using Equation 2.3. That is, estimated as the fraction, over P, of the number of datasets $\tilde{\mathcal{D}}^{(j)}$ that contain at least one sequential pattern with p-value $\leq \delta$, and thus a sequential pattern that would be reported as significant while it is not when δ is used as significance threshold. Since ProMiSe uses the corrected significance threshold $\delta^* = \max\{\delta : \text{FWER}(\delta) \leq \alpha\}$, then its output has FWER $\leq \alpha$, which concludes the proof. □

Let us note that ProMiSe crucially hinges on the ability of sampling random datasets with the properties described above.

6.3.1 Efficiently Sampling Random Datasets

In this Section, we describe three methods to obtain a random dataset $\tilde{\mathcal{D}}$ where:

- each itemset appears the same number of times it appears in $\mathcal{D}$;
- each transaction has the same length it has in $\mathcal{D}$.

All three strategies, i.e., *itemsets swapping*, *dataset permutation*, and *transactions permutations*, start from $\mathcal{D}$ and perform random operations (*swaps* or *permutations*) at the level of itemsets. While itemsets swapping and dataset permutation only preserve the two properties above, transactions permutations also focuses on preserving additional properties of $\mathcal{D}$, which will be described in the respective Section, allowing different types of analyses. Let us remember that sequential patterns describe ordered sequences of events (i.e., itemsets), and thus the basic idea of all these strategies is to preserve these events, represented by the itemsets, and to only change the order in which they occur.

Itemsets Swapping

We now describe a strategy similar to permutation swapping [13], previously proposed for significant itemsets mining [21]. Let us note that the original strategy proposed in [21] cannot be directly used for sequential patterns since it does not take into account the order of the itemsets and an itemsets can appear more than once in a sequential transaction.

For each transaction $\tau_i \in \mathcal{D}$, $i \in \{1, \ldots, |\mathcal{D}|\}$, and each integer $j \in \{1, \ldots, |\tau_i|\}$, we use the pair (i, j) to represent the the j-th itemset in τ_i. In *itemset swapping (IS)*, we swap the itemset in position (i, j) with the itemset in position (k, ℓ), with $i, k \in \{1, \ldots, |\mathcal{D}|\}$, $j \in \{1, \ldots, |\tau_i|\}$ and $\ell \in \{1, \ldots, |\tau_k|\}$. Let us note that such a swap preserves the length of each transaction while the item-lengths of the transactions are not necessary preserved, since the size of the two swapped itemsets may not be the same. Also the frequencies of the swapped itemsets can change after a swap since the new transaction could contain itemsets that are super-sets of the swapped ones. Finally, for the same reason, also the frequencies of the items that compose the two itemsets can change as well.

Example 7. *Let us consider the dataset shown in Example 6. (Here, we refer to it as dataset $\mathcal{D}$). In such a dataset, a possible swap is performed between itemset $\{3\}$ in position $(1, 1)$ and itemset $\{6, 7\}$ in position $(2, 3)$. The new dataset $\tilde{\mathcal{D}}$ after the swap is the following:*

$$\begin{aligned}
\tau_1 &= \langle\{6, 7\}, \{2, 6, 7\}\rangle \\
\tau_2 &= \langle\{7\}, \{2\}, \{3\}, \{1, 2, 5, 6\}\rangle \\
\tau_3 &= \langle\{6, 7\}, \{2\}, \{2\}, \{2\}\rangle \\
\tau_4 &= \langle\{1\}, \{2\}, \{1, 4\}, \{2\}\rangle.
\end{aligned}$$

Let us note that the length of the transactions τ_1 and τ_2 does not change after the swap, contrary to their item-length. The frequency of $\{3\}$ has remained the same, $f_{\mathcal{D}}(\{3\}) = f_{\tilde{\mathcal{D}}}(\{3\}) = 1/4$, while the frequency of $\{6, 7\}$ has changed, $f_{\mathcal{D}}(\{6, 7\}) = 3/4 \neq 1/2 = f_{\tilde{\mathcal{D}}}(\{6, 7\})$.

Let us note that in a dataset $\mathcal{D}$ there are $m = \sum_{i=1}^{|\mathcal{D}|} |\tau_i|$ total itemsets (not necessarily all distinct). Therefore, we can use the integer $\ell \in \{1, \ldots, m\}$ to identify each itemset. Algorithm 9 shows how to generate a random dataset using a sequence of r itemsets swap operations under this indexing of itemsets. The procedure `Swap`$(\mathcal{D}, p_1, p_2)$ simply swaps the itemset of index p_1 with the itemset of index p_2 in $\mathcal{D}$, while the procedure `Random`$(1, m)$ generates an integer uniformly at random between 1 and m.

We now prove that the dataset produced in output by Algorithm 9 is a dataset taken uniformly at random among the set $\mathbb{D}$ of all datasets that satisfy the properties described in Section 6.3.1, providing enough swap operations. To analyze Algorithm 9, we use the Markov chains framework [52]. Let us consider the Markov chain $C = \{\mathcal{P}, \mathcal{T}\}$, where

$\mathcal{P}$ is the *state space* and $\mathcal{T}$ is the set of *transitions*. In our case, $\mathcal{P}$ is the set $\mathbb{D}$ of all datasets satisfying the two properties defined in Section 6.3.1, while $\mathcal{T}$ is defined in terms of *neighbors* of a dataset $\mathcal{D}_j \in \mathbb{D}$, where the neighbors of $\mathcal{D}_j$ are obtained by performing a single swap operation on $\mathcal{D}_j$. That is, the set $\mathcal{T}$ contains all pairs with datasets $(\mathcal{D}_j, \mathcal{D}_k)$ s.t. it is possible to obtain $\mathcal{D}_k$ from $\mathcal{D}_j$ (or vice versa) with a single itemsets swap, with $\mathcal{D}_j, \mathcal{D}_k \in \mathbb{D}$. For each dataset (state) $\mathcal{D}_j \in \mathbb{D}$, we define the *degree* of state $\mathcal{D}_j$ in the Markov chain C as the number of possible swaps that can be performed in dataset $\mathcal{D}_j$.

To prove that the output of Algorithm 9 produces in output a random dataset starting from a dataset $\mathcal{D}$, we prove that the Markov chain C admits a unique stationary distribution and that such a distribution is uniform among all elements of $\mathbb{D}$.

Algorithm 9: `RandomDatasetIS`: generate a random dataset using the itemsets swapping strategy.

```
**Data:** Sequential dataset $\mathcal{D} = \{\tau_1, \tau_2, \dots, \tau_{|\mathcal{D}|}\}$, number of itemsets swaps $r$.
**Result:** Random dataset $\tilde{\mathcal{D}} = \{\tau'_1, \tau'_2, \dots, \tau'_{|\tilde{\mathcal{D}}|}\}$.
1 $\tilde{\mathcal{D}} \leftarrow \mathcal{D}$;
2 $m \leftarrow \sum_{i=1}^{|\mathcal{D}|} |\tau_i|$;
3 for $i \leftarrow 1$ to $r$ do
4     $p_1 \leftarrow$ Random$(1, m)$;
5     $p_2 \leftarrow$ Random$(1, m)$;
6     Swap$(\tilde{\mathcal{D}}, p_1, p_2)$;
7 return $\tilde{\mathcal{D}}$;
```

Theorem 19. *The Markov chain C admits as unique stationary distribution the uniform distribution.*

Proof. Let us note that starting from any dataset $\mathcal{D}_j \in \mathbb{D}$, it is possible to obtain any other dataset $\mathcal{D}_k \in \mathbb{D}$ with a series of itemsets swap operations, that is, the Markov chain C is irreducible. The Markov chain C has a finite state space, it is irreducible, and it is aperiodic. A Markov chain with these properties is an ergodic chain. The Markov chain C is also reversible: an itemsets swap can be undone by a single (reversed) itemsets swap. From the theory of Markov chains [52], an ergodic Markov chain has a unique stationary distribution. From the reversibility property, it follows that the probability of each state in such a stationary distribution is proportional to the degree of the states. Therefore, in order to obtain a uniform distribution, all states of the Markov chain must have the same degree. Using Algorithm 9 to generate dataset $\tilde{\mathcal{D}}$, all the states of the Markov chain have degree equals to m^2, with $m = \sum_{i=1}^{|\mathcal{D}|} |\tau_i|$. Thus, the Markov chain C admits as unique stationary distribution the uniform distribution, which concludes the proof. □

In order to bound the number of swap operations that are required to converge to the stationary distribution, we need to upper bound the *mixing time* of the Markov chain C. Upper bounding the mixing time is usually difficult. For example, the mixing time of the commonly used swap randomization procedure [13, 22] has been the object of theoretical studies [13], but currently there are no conclusive results and only empirical analyses are available [50, 21].

We now prove an upper bound on the number of itemsets swap operations that are required for the Markov chain C to converge to the stationary distribution. In particular, we aim to compute an upper bound on the *mixing time* $\eta(\varepsilon)$ which guarantees that for any initial state of C, the variation distance between the distribution of its states after $\eta(\varepsilon)$ steps and its stationary distribution is at most $\varepsilon \in (0,1]$. In our proof, we use the *path coupling* technique [52]. In brief, given a Markov chain C, a coupling for C consists of two copies, C_1 and C_2, of C running simultaneously, where the two copies do not visit the states in the same order nor perform the same transition at the same time, but are defined on the same state space of C and have the same transition probabilities as C.

Theorem 20. *The mixing time $\eta(\varepsilon)$ of the Markov chain C is $\mathcal{O}(m^2 \log \frac{m}{\varepsilon})$, where $m = \sum_{i=1}^{|\mathcal{D}|} |\tau_i|$ and $\varepsilon \in (0,1]$.*

Proof (sketch). We use path coupling to prove that the difference between the distribution of the states of the Markov chain C after $\mathcal{O}(m^2 \log \frac{m}{\varepsilon})$ itemsets swaps and its stationary distribution is at most ε. Let $\mathcal{D}_\ell, \mathcal{D}_h \in \mathbb{D}$ be two datasets that differ only for the position of two itemsets. We say that $\mathcal{D}_\ell$ and $\mathcal{D}_h$ are at distance $dist(\mathcal{D}_\ell, \mathcal{D}_h) = 2$. The idea is to start with a coupling for such a pair of datasets, that differ in just two itemsets, and then extend the coupling over all pairs of datasets. Denoting with a and b the two positions where datasets $\mathcal{D}_\ell$ and $\mathcal{D}_h$ differ, we define a coupling where the first Markov chain C_1 is exactly C, while the second Markov chain C_2 is defined in terms of the transitions of C_1. Let $\mathcal{D}_\ell$ be the state of $C_1 = C$ at a given iteration, and let $\mathcal{D}_h$ be the state of C_2 at the same iteration, i.e., the two Markov chains are in the two datasets at distance $dist(\mathcal{D}_\ell, \mathcal{D}_h) = 2$. The transition $(\mathcal{D}_\ell, \mathcal{D}_\ell')$ performed by C_1 (from state $\mathcal{D}_\ell$) consists of an itemsets swap between the itemset in position p_1 and the itemset in position p_2 in dataset $\mathcal{D}_\ell$, with p_1 and p_2 sampled uniformly at random among all m possible positions. We then define the transition $(\mathcal{D}_h, \mathcal{D}_h')$ for C_2 (from state $\mathcal{D}_h$) as follows:

1. if $p_1 = a$ and $p_2 = b$, C_2 swaps the itemset in position a with itself in the dataset $\mathcal{D}_h$;
2. if $p_1 = b$ and $p_2 = a$, C_2 swaps the itemset in position b with itself in the dataset $\mathcal{D}_h$;

3. if $p_1 = p_2 = a$ or $p_1 = p_2 = b$, C_2 swaps the itemset in position a with the itemset in position b in the dataset $\mathcal{D}_h$;
4. otherwise C_2 swaps the itemset in position p_1 with the itemset in position p_2 in the dataset $\mathcal{D}_h$.

Let us note that for both chains, the probability of any given transition is still $1/m^2$ and thus the coupling is valid. Most of the moves of such a coupling maintain the distance $dist(\mathcal{D}'_\ell, \mathcal{D}'_h) = 2$. The only moves that result in $dist(\mathcal{D}'_\ell, \mathcal{D}'_h) = 0$ are the 4 moves described by 1-3, i.e., the moves that swap, in one of the two datasets, the two itemsets that are in different positions. Since each move occurs with the same probability $1/m^2$ and $dist(\mathcal{D}_\ell, \mathcal{D}_h) = 2$, we have

$$\begin{aligned}\mathbb{E}\left[dist(\mathcal{D}'_\ell, \mathcal{D}'_h)|\mathcal{D}_\ell, \mathcal{D}_h\right] &= \left(1 - \tfrac{4}{m^2}\right) \cdot 2 + \tfrac{4}{m^2} \cdot 0 \\ &\leq (1-\beta) \cdot dist(\mathcal{D}_\ell, \mathcal{D}_h),\end{aligned}$$

with $\beta \leq 4/m^2$. Thus, by applying the path coupling theorem [52], we can extend the coupling to arbitrary pairs of states $(\mathcal{D}^0_\ell, \mathcal{D}^0_h)$ (at any distance) obtaining a bound on the mixing time of $\eta(\varepsilon) = \mathcal{O}(\frac{1}{\beta} \log \frac{D}{\varepsilon})$, where D is the maximum distance between any two states. Since in our case we have $D = m$ and $\beta \leq 4/m^2$, we obtain that the mixing time of C is bounded by $\eta(\varepsilon) = \mathcal{O}(m^2 \log \frac{m}{\varepsilon})$, which concludes the proof. $\square$

Theorems 19 and 20 tell us that Algorithm 9 can be employ to sample uniformly at random a dataset among the set $\mathbb{D}$ if enough itemsets swap operations are provided, and give us an upper bound on the number of swaps to perform. Moreover, in Section 6.5, we experimentally show that a number of swaps proportional to the number of itemsets in the dataset is sufficient to sample a random dataset. In addition, further studies allowed us to define a more efficient strategy, i.e., dataset permutation, to generate datasets from the aforementioned null hypothesis avoiding the series of itemsets swaps.

Dataset Permutation

In this Section, we introduce an alternative strategy, *dataset permutation (DP)*, to sample a dataset uniformly at random among the set $\mathbb{D}$ of all datasets that satisfy the properties described in Section 6.3.1. Let $\mathcal{L}(\mathcal{D})$ be the *ordered sequence* of itemsets obtained concatenating all the transactions of $\mathcal{D} = \{\tau_1, \tau_2, \ldots, \tau_{|\mathcal{D}|}\}$, starting from the first one, τ_1, and ending with the last one, $\tau_{|\mathcal{D}|}$. Let us note that at each dataset corresponds a unique ordered sequence constructed in such a way. The idea behind the dataset permutation strategy is the following. Starting from a dataset $\mathcal{D}$, we create its ordered sequence

$\mathcal{L}(\mathcal{D})$. Performing a random permutation of $\mathcal{L}(\mathcal{D})$, we obtain a new ordered sequence $\mathcal{L}(\tilde{\mathcal{D}})$ which contains the same itemsets of $\mathcal{L}(\mathcal{D})$ but in a different order. Finally, starting from $\mathcal{L}(\tilde{\mathcal{D}})$, we construct the random dataset $\tilde{\mathcal{D}} = \{\tau_1', \tau_2', \dots, \tau_{|\mathcal{D}|}'\}$ associated with $\mathcal{L}(\tilde{\mathcal{D}})$. That is, the first $|\tau_1|$ itemsets of $\mathcal{L}(\tilde{\mathcal{D}})$ form the transaction τ_1', the following $|\tau_2|$ itemsets of $\mathcal{L}(\tilde{\mathcal{D}})$ form the transaction τ_2', and so on. Since at each dataset corresponds a unique ordered sequence, if we generate a random permutation uniformly at random among all possible permutations, then we sample a dataset uniformly at random among $\mathbb{D}$. Such a permutation can be generated, for example, with Algorithm P (Shuffling) [37], which takes time proportional to the number of itemsets to permute, i.e., $\mathcal{O}(m)$, and shuffles them in place. Thus, the dataset permutation strategy is more efficient than the itemsets swapping one.

Example 8. *Let us consider again the dataset shown in Example 6. (Here, we refer to it as dataset $\mathcal{D}$). The ordered sequence $\mathcal{L}(\mathcal{D})$ associated with such a dataset is*

$$\mathcal{L}(\mathcal{D}) = \langle \{3\}, \{2,6,7\}, \{7\}, \{2\}, \{6,7\}, \{1,2,5,6\}, \{6,7\}, \{2\}, \{2\}, \{2\}, \{1\}, \{2\}, \{1,4\}, \{2\} \rangle.$$

A possible permutation of such an ordered sequence is

$$\mathcal{L}(\tilde{\mathcal{D}}) = \langle \{2\}, \{7\}, \{2\}, \{3\}, \{2,6,7\}, \{6,7\}, \{1\}, \{2\}, \{6,7\}, \{2\}, \{2\}, \{2\}, \{1,2,5,6\}, \{1,4\} \rangle.$$

Finally, the random dataset $\tilde{\mathcal{D}}$ associated with $\mathcal{L}(\tilde{\mathcal{D}})$ is

$$\begin{aligned}
\tau_1 &= \langle \{2\}, \{7\} \rangle \\
\tau_2 &= \langle \{2\}, \{3\}, \{2,6,7\}, \{6,7\} \rangle \\
\tau_3 &= \langle \{1\}, \{2\}, \{6,7\}, \{2\} \rangle \\
\tau_4 &= \langle \{2\}, \{2\}, \{1,2,5,6\}, \{1,4\} \rangle.
\end{aligned}$$

Let us note that the same exact considerations done in Example 7 are valid also for this resulting random dataset.

The dataset permutation strategy is described in Algorithm 10.[2]

Transactions Permutations

We now introduce a different strategy to obtain a random dataset, *transactions permutations (TP)*, which permutes each transaction independently. Let us note that this strategy produces a random dataset that still satisfies the properties described above, but

[2] $+\!\!+$ in Algorithm 10 is the concatenation of an element to a sequence.

Algorithm 10: `RandomDatasetDP`: generates a random dataset with the dataset permutation strategy.

```
  Data: Sequential dataset $\mathcal{D} = \{\tau_1, \tau_2, \ldots, \tau_{|\mathcal{D}|}\}$.
  Result: Random dataset $\tilde{\mathcal{D}} = \{\tau'_1, \tau'_2, \ldots, \tau'_{|\tilde{\mathcal{D}}|}\}$.
1 $\mathcal{L}(\mathcal{D}) \leftarrow$ Empty sequence;
2 for $i \leftarrow 1$ to $|\mathcal{D}|$ do
3     $\mathcal{L}(\mathcal{D}) \leftarrow \mathcal{L}(\mathcal{D}) \mathbin{+\!\!+} \tau_i$;
4 $\mathcal{L}(\tilde{\mathcal{D}}) \leftarrow$ Permute($\mathcal{L}(\mathcal{D})$);
5 $\tilde{\mathcal{D}} \leftarrow$ Dataset associated with $\mathcal{L}(\tilde{\mathcal{D}})$;
6 return $\tilde{\mathcal{D}}$;
```

it also forces the itemsets to appear in the same transactions as in the original dataset $\mathcal{D}$. In particular, this strategy ensures that only the *order* with which the itemsets appear in each transaction is random, while everything else (e.g., itemsets frequencies, transactions in which each itemset appears, etc.) is fixed.

As a motivation, let us consider a dataset containing movies rated by some users. In such a dataset, the items are the ID's of the movies rated by the users. Each transaction contains the ID's of the movies rated by a single user and movies rated in the same temporal interval, i.e., in the same day, are grouped in a single itemset. Thus, the transactions represent the temporal sequence of sets of movies rated in the same temporal interval. Since a user usually rates a movie only once, a movie is present at most once in each transaction. This feature of the data is not preserved by itemsets swapping, nor by dataset permutation. Thus, using itemsets swapping or dataset permutation in dataset which exhibits this property, ProMiSe would report many significant sequential patterns that are not interesting in such a scenario, since the strategy used to generate random datasets completely change the original distribution of the data.

Example 9. *Let us consider again the dataset shown in Example 6. (Here, we refer to it as dataset $\mathcal{D}$). A random dataset $\tilde{\mathcal{D}}$ generated from $\mathcal{D}$ using the transactions permutations approach is the following:*

$$\begin{aligned}
\tau_1 &= \langle \{3\}, \{2, 6, 7\} \rangle \\
\tau_2 &= \langle \{2\}, \{1, 2, 5, 6\}, \{6, 7\}, \{7\} \rangle \\
\tau_3 &= \langle \{2\}, \{2\}, \{6, 7\}, \{2\} \rangle \\
\tau_4 &= \langle \{1, 4\}, \{1\}, \{2\}, \{2\} \rangle.
\end{aligned}$$

Let us note that each transaction of the new dataset has the same length, and also the same item-length, of the respective transaction of the dataset $\mathcal{D}$. Also the frequency *of the items and of the itemsets remains the same in the two datasets. Instead, the frequency of the sequential patterns can change:* $f_{\mathcal{D}}(\langle\{3\},\{2\}\rangle) = f_{\tilde{\mathcal{D}}}(\langle\{3\},\{2\}\rangle) = 1/4$ *but* $f_{\mathcal{D}}(\langle\{2\},\{6,7\}\rangle) = 1/4 \neq 1/2 = f_{\tilde{\mathcal{D}}}(\langle\{2\},\{6,7\}\rangle)$.

The transactions permutations strategy, described in Algorithm 11, produces a random dataset by permuting each transaction independently of all other events.[3] Thus, it samples a random dataset uniformly at random among all datasets with the desired properties.

Algorithm 11: `RandomDatasetTP`: generate a random dataset with the transactions permutations strategy.

```
Data: Sequential dataset $\mathcal{D} = \{\tau_1, \tau_2, \ldots, \tau_{|\mathcal{D}|}\}$.
Result: Random dataset $\tilde{\mathcal{D}} = \{\tau'_1, \tau'_2, \ldots, \tau'_{|\tilde{\mathcal{D}}|}\}$.
1 for $i \leftarrow 1$ to $|\mathcal{D}|$ do
2     $\tau'_i \leftarrow$ Permute($\tau_i$);
3     $\tilde{\mathcal{D}} \leftarrow \tilde{\mathcal{D}} \uplus \tau'_i$;
4 return $\tilde{\mathcal{D}}$;
```

6.3.2 Parallel Implementation

Let us note that ProMiSe, as all approaches based on permutation testing, is well-suited to parallelization. In particular, the generation of random datasets and the computation of the p-values for the sequential patterns in $FSP(\mathcal{D},\theta)$ can be easily parallelized: when k cores are used to compute the p-values using M random datasets in the MC estimate, each core computes the p-values on M/k random datasets, and the results are then aggregated at the end. Thus, for each core and for each pattern $s \in FSP(\mathcal{D},\theta)$, it computes the sum of the values $\mathbb{1}[f_{\mathcal{D}_i}(s) \geq f_{\mathcal{D}}(s)]$, considering sequentially M/k random datasets $\mathcal{D}_i$. Finally, it aggregates the counts computed by each core, using Equation 6.1, obtaining the p-value p_s.

Such a parallel implementation is particularly advantageous for ProMiSe, since the MC estimate of the p-values of the sequential patterns is also required by the WY method that computes, for each random dataset $\tilde{\mathcal{D}}^{(j)}$, with $j \in \{1,\ldots,P\}$, the minimum observed p-value in the dataset $\tilde{\mathcal{D}}^{(j)}$, over all sequential patterns of interest, i.e., the ones with $f_{\tilde{\mathcal{D}}^{(j)}}(s) \geq \theta$.

[3] $\uplus$ in Algorithm 11 is the addition of an element to a multi-set.

6.4 I-ProMiSe: Mining Significant Sequential Patterns with Confidence Intervals

In this Section, we describe I-ProMiSe, an alternative version of ProMiSe which uses confidence Intervals. Let us note that ProMiSe requires the generation of a large amount of random datasets for the MC procedure, for the original dataset and for the P random datasets required by the WY method. Even if in the previous Sections we introduced efficient strategies to generate random datasets, this could be computationally infeasible for massive datasets. For this reason, we introduce I-ProMiSe, a version of ProMiSe which results in a lower statistical power than the original version but that provides a more efficient solution to mine significant sequential patterns from massive datasets. Let us remember that in significant pattern mining, the dataset is seen as a sample from an unknown distribution and one is interested in finding patterns significantly deviating from an assumed null distribution. Thus, we can consider that the dataset $\mathcal{D}$ is a sample from the *unknown distribution* π_U and we are interested in mining sequential patterns s which have true frequency $t_{\pi_U}(s)$ w.r.t. π_U greater than their true frequency $t_{\pi_N}(s)$ w.r.t. the *null distribution* π_N, that is, sequential patterns s with $t_{\pi_U}(s) > t_{\pi_N}(s)$. Since, from the results of the previous Sections, we are able to sample datasets from the null distribution, we can use an approach analogous of the one introduced in Chapter 5 to mine emerging patterns in (a sequence of) two datasets to mine statistically significant sequential patterns. (See Section 5.4.1 for more details.)

The idea behind I-ProMiSe is the following. It starts by generating a single random dataset $\tilde{\mathcal{D}}$. Let us remember that, given a sequential dataset $\mathcal{D}$ that is a sample from a probability distribution π, the maximum deviation is defined as $\sup_{s\in\mathbb{S}} |t_\pi(s) - f_{\mathcal{D}}(s)|$. Thus, I-ProMiSe computes an upper bound on the maximum deviation for the null distribution π_N in the random dataset $\tilde{\mathcal{D}}$ and for the unknown distribution π_U in the original dataset $\mathcal{D}$. Finally, it uses such upper bounds on the maximum deviation to find sequential patterns $s \in FSP(\mathcal{D}, \theta)$ s.t. $t_{\pi_U}(s) > t_{\pi_N}(s)$, providing guarantees on the FWER.

Algorithm 12 shows the pseudo-code of I-ProMiSe. Given a sequential dataset $\mathcal{D}$, a minimum frequency threshold $\theta \in (0, 1]$, and a FWER threshold $\alpha \in (0, 1)$ as input, I-ProMiSe identifies a set of significant sequential patterns with frequency at least θ and FWER bounded by α. I-ProMiSe starts by mining the set $FSP(\mathcal{D}, \theta)$ of sequential patterns with frequency at least θ in $\mathcal{D}$ (lines 2). As already said for ProMiSe, the extraction of such sequential patterns can be performed with any efficient algorithm for the mining of frequent sequential patterns. Then, it generates a random dataset $\tilde{\mathcal{D}}$ (line 3) using one of the procedure defined in the previous Sections, i.e., Algorithm 9, 10 or 11. From the dataset $\mathcal{D}$, it computes a probabilistic upper bound μ_U on the maximum deviation $\sup_{s\in\mathbb{S}} |t_{\pi_U}(s) - f_{\mathcal{D}}(s)|$ (line 4), while from the random dataset $\tilde{\mathcal{D}}$, it computes a probabilistic upper bound μ_N on the maximum deviation $\sup_{s\in\mathbb{S}} |t_{\pi_N}(s) - f_{\tilde{\mathcal{D}}}(s)|$ (line 5), both using $\alpha/2$ as confidence parameter. (Such upper bounds can be computed, for example, using the VC-dimension as described in the previous Chapter, Section 5.5.) Finally, it reports in output all the sequential patterns $s \in FSP(\mathcal{D},\theta)$ with $f_{\mathcal{D}}(s) > f_{\tilde{\mathcal{D}}}(s) + \mu_U + \mu_N$ (line 6-9). Let us note that only if $f_{\mathcal{D}}(s) > f_{\tilde{\mathcal{D}}}(s) + \mu_U + \mu_N$, we can be sure that $t_{\pi_U}(s) > t_{\pi_N}(s)$, avoiding false positives.

Algorithm 12: I-ProMiSe

```
  Data: Sequential dataset $\mathcal{D}$, minimum frequency threshold $\theta \in (0,1]$, FWER
        threshold $\alpha \in (0,1)$.
  Result: Set $\mathcal{SSP}$ of significant sequential patterns with FWER $\le \alpha$.
1 $\mathcal{SSP} \leftarrow \emptyset$;
2 $\mathcal{F} \leftarrow FSP(\mathcal{D},\theta)$;
3 $\tilde{\mathcal{D}} \leftarrow$ RandomDataset($\mathcal{D}$);
4 $\mu_U \leftarrow$ ComputeMaxDev($\mathcal{D}, \alpha/2$);
5 $\mu_N \leftarrow$ ComputeMaxDev($\tilde{\mathcal{D}}, \alpha/2$);
6 foreach $s \in \mathcal{F}$ do
7     if $f_{\mathcal{D}}(s) > f_{\tilde{\mathcal{D}}}(s) + \mu_U + \mu_N$ then
8         $\mathcal{SSP} \leftarrow \mathcal{SSP} \cup s$;
9 return $\mathcal{SSP}$;
```

The following result establishes the quality of the output of I-ProMiSe.

Theorem 21. *The output of* I-ProMiSe *has FWER* $\le \alpha$.

Proof. Let us consider that $D(\mathcal{D},\pi_U) = \sup_{s\in\mathbb{S}} |t_{\pi_U}(s) - f_{\mathcal{D}}(s)| \le \mu_U$ and that $D(\tilde{\mathcal{D}},\pi_N) = \sup_{s\in\mathbb{S}} |t_{\pi_N}(s) - f_{\tilde{\mathcal{D}}}(s)| \le \mu_N$. Let s' be a sequential pattern s.t. $t_{\pi_U}(s') \le t_{\pi_N}(s')$. Since we know that $t_{\pi_U}(s') \in [f_{\mathcal{D}}(s') - \mu_U, f_{\mathcal{D}}(s') + \mu_U]$ and that $t_{\pi_N}(s') \in [f_{\tilde{\mathcal{D}}}(s') - \mu_N, f_{\tilde{\mathcal{D}}}(s') + \mu_N]$, then $f_{\mathcal{D}}(s) - \mu_U \le f_{\tilde{\mathcal{D}}}(s) + \mu_N$ and thus s' cannot be reported in output by I-ProMiSe. Since from the confidence parameters used to compute the upper bounds on the maximum deviation, $\Pr(D(\mathcal{D},\pi_U) \le \mu_U) \ge 1 - \alpha/2$ and $\Pr(D(\tilde{\mathcal{D}},\pi_N) \le \mu_N) \ge 1 - \alpha/2$, then the output of I-ProMiSe has FWER $\le \alpha$, which concludes the proof. □

6.5 Experimental Evaluation

In this Section, we report the results of our experimental evaluation on multiple pseudo-artificial and real datasets to assess the performance of ProMiSe and I-ProMiSe for mining statistically significant sequential patterns.

The goals of the evaluation are the following:

- To empirically estimate the number of itemsets swaps needed to reach the stationary distribution for the itemsets swapping strategy (see Section 6.3.1).
- To prove that ProMiSe is able to find large sets of statistically significant sequential patterns in real datasets, while avoiding false positives.
- To make a comparison between ProMiSe and I-ProMiSe for mining significant sequential patterns from real datasets.

6.5.1 Implementation, Environment, and Datasets

We implemented ProMiSe and I-ProMiSe in Java, and used Apache Spark Java API version 3.1.1 to parallelize ProMiSe. To mine the frequent sequential patterns, we used the PrefixSpan [54] implementation provided by the SPMF library [18]. To bound the maximum deviation as required by I-ProMiSe, we used the results of Chapter 5 based on the empirical VC-dimension of sequential patterns (see Section 5.5). We performed all the experiments on the same machine with 512 GB of RAM and 2 Intel(R) Xeon(R) CPU E5-2698 v3 @ 2.3GHz with a total of 64 cores, running Ubuntu 14.14 and using Java 1.8.0_201. Our open-source parallel implementation of ProMiSe, our implementation of I-ProMiSe, and the code developed for the tests are available at https://github.com/VandinLab/PROMISE. In all the experiments, we fixed the FWER threshold to the commonly used value $\alpha = 0.05$ and we used 64 cores to parallelize the execution of ProMiSe. We now describe the datasets used in the evaluation, and how we generated them, when necessary. Table 6.1 shows their characteristics.

- BIBLE:[4] a conversion of the Bible into a sequence dataset. A word is an item and a sentence corresponds to a transaction.
- BIKE:[5] Los Angeles Metro Bike Share Trip Data. An item is a bike-sharing station and a transaction is the sequence of bike-sharing stations in which a given bike was. Details about the generation of such a dataset are provided in Section 6.5.6.

[4]http://www.philippe-fournier-viger.com/spmf/index.php?link=datasets.php
[5]https://www.kaggle.com/cityofLA/los-angeles-metro-bike-share-trip-data

Table 6.1 Datasets characteristics. The table reports: Dataset $\mathcal{D}$: name of the dataset; $|\mathcal{D}|$: number of transactions; $|\mathcal{I}|$: total number of items; Avg $||\tau||$: average transaction item-length; Rep items: whether the items appear multiple time in a transaction.

Dataset $\mathcal{D}$	$\|\mathcal{D}\|$	$\|\mathcal{I}\|$	Avg. $\|\|\tau\|\|$	Rep. items
BIBLE	36369	13905	21.6	Yes
BIKE	21078	67	7.28	Yes
FIFA	20450	2990	36.2	Yes
FLIGHT	151M	523	2.77	Yes
FLIGHT1	78.5M	503	2.87	Yes
FLIGHT2	87.3M	489	2.69	Yes
LEVIATHAN	5835	9025	33.8	Yes
SIGN	730	267	52.0	No

- FIFA:[4] a dataset of sequences of click-stream data from the website of FIFA World Cup 98. An item represents a web page and the sequence of click-steams of a single session of a user is a transaction.

- FLIGHT:[6] data of the commercial flights in the USA. Each item is an airport, while each transaction represents the sequence of airports visited in a single itinerary by a passenger. We considered the data of the "Origin and Destination Survey: DB1BCoupon", between 2010 and 2020. Such data contains single flights in the format of origin and destination airports, an unique numerical identifier of the itinerary that contains the flight, and the sequence number of the flight inside the itinerary. We collected the temporal ordered sequence of airports that each passenger visited in a single itinerary, sorting the airports using the sequence numbers. Such a sequence is a transaction in our dataset.

- FLIGHT1: smaller version of FLIGHT that contains data between 2010 and 2015.

- FLIGHT2: smaller version of FLIGHT that contains data between 2015 and 2020.

- LEVIATHAN:[4] this dataset is a conversion of the novel Leviathan by Thomas Hobbes (1651) as a sequence database. A word is an item and a sentence corresponds to a transaction.

- SIGN:[4] a dataset of sign language utterance.

[6]https://www.transtats.bts.gov/Fields.asp?gnoyr_VQ=FLM

6.5.2 Itemsets Swaps Convergence

In this Section, we empirically study the number of itemsets swaps needed to reach the stationary distribution, that is, the number of itemsets swaps to perform in order to obtain a dataset drawn uniformly at random. In this study, we did not consider the three FLIGHT datasets for the excessive computational time required to generate their random datasets. Theorem 20 gives us an upper bound of the number of swaps to perform. In practice, the stationary distribution may be reached with a smaller number of swaps. To evaluate whether this is the case, we analyzed how the *average relative frequency difference (ARFD)* changed with the number of itemsets swaps.

Given the real dataset $\mathcal{D}$, let the set $FSP(\mathcal{D},\theta)$ be the frequent sequential patterns extracted from $\mathcal{D}$ using θ as minimum frequency threshold, and let $\tilde{\mathcal{D}}_t$ be a random dataset obtained performing a series of t itemsets swaps from $\mathcal{D}$. We define the *average relative frequency difference* $ARFD(\tilde{\mathcal{D}}_t)$ for $\tilde{\mathcal{D}}_t$ as

$$ARFD(\tilde{\mathcal{D}}_t) = \frac{1}{|FSP(\mathcal{D},\theta)|} \sum_{s \in FSP(\mathcal{D},\theta)} \frac{|f_{\mathcal{D}}(s) - f_{\tilde{\mathcal{D}}_t}(s)|}{f_{\mathcal{D}}(s)}.$$

The idea behind the $ARFD$ is to observe how the frequency, and so the distribution, of the frequent sequential patterns of $\mathcal{D}$ changes after a series of t random swaps. If $ARFD(\tilde{\mathcal{D}}_t)$ does not change much after $t > t^*$ swaps for some value t^*, then the distribution of the sequential patterns does not vary much in the datasets generated with more than t^* swaps, and we can assume that we are close to the stationary distribution. We computed $ARFD(\tilde{\mathcal{D}}_t)$ for $t = im$, where i is a positive integer and $m = \sum_{i=1}^{|\mathcal{D}|} |\tau_i|$. For each real dataset, we performed this computation 5 times (i.e., generating five different random datasets $\tilde{\mathcal{D}}_t$, each with a series of t itemsets swaps, starting from $\mathcal{D}$) and then we computed the average value of $ARFD(\tilde{\mathcal{D}}_{im})$. The dashed lines shown in Figure 6.1 are the values obtained for all the real datasets. Let us note that we performed a maximum of $m^2 \log m$ itemsets swaps for all the datasets.

From Figure 6.1, it is possible to notice that, just after $2m$ itemsets swaps and for all the datasets, the $ARFD(\tilde{\mathcal{D}}_{2m})$ is very close to the one obtained with $m^2 \log(m)$ itemsets swaps. Thus, we considered that $2m$ swaps are enough to sample a random datasets uniformly at random. Finally, we generated a single random dataset $\tilde{\mathcal{D}}$ using the dataset permutation strategy, we computed $ARFD(\tilde{\mathcal{D}})$, and we compared it with the one obtained with itemsets swaps. Again, we repeated the computation 5 times and we reported the average. The solid lines in Figure 6.1 show the results. It is possible to notice that $ARFD(\tilde{\mathcal{D}})$, obtained with a single dataset permutation of $\mathcal{D}$, is very close to

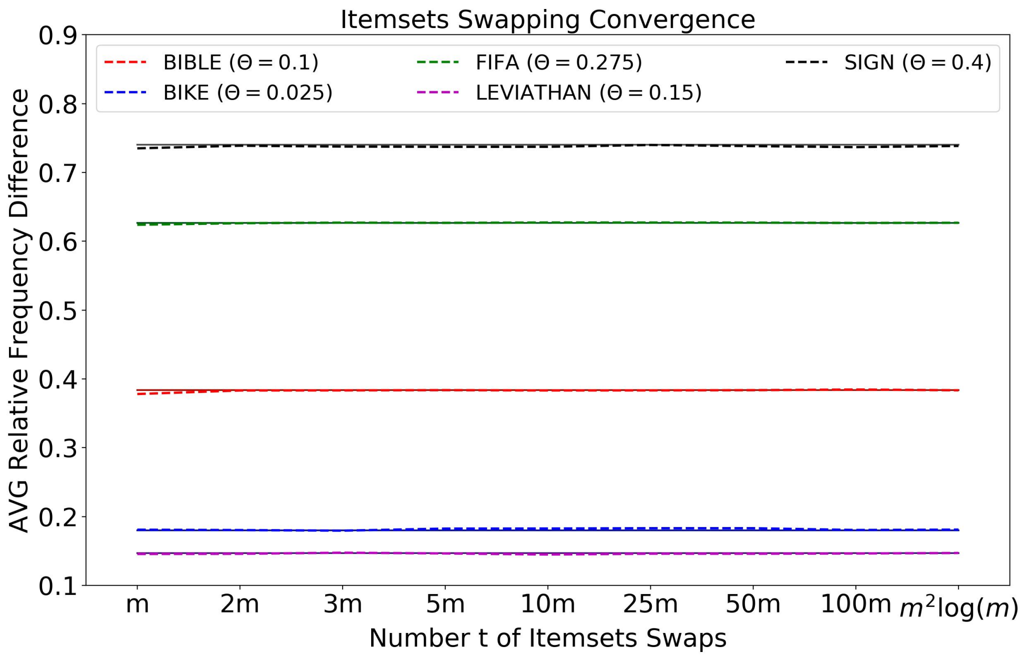

Fig. 6.1 Itemsets Swapping Convergence. It shows: average relative difference $ARFD(\tilde{\mathcal{D}}_t)$ between the frequencies of the frequent sequential patterns in $\mathcal{D}$ and their frequencies in random datasets generated with t itemsets swaps (dashed lines); average relative difference $ARFD(\tilde{\mathcal{D}})$ between the frequencies of the frequent sequential patterns in $\mathcal{D}$ and their frequencies a random datasets generated with the dataset permutation strategy (solid lines). The legend shows the values of θ used to compute the $ARFD$.

the one obtained with the itemsets swaps (in particular with $m^2\log(m)$ itemsets swaps), and thus the dataset permutation is a valid and more efficient strategy to sample datasets uniformly at random from the null distribution.

Let us note that the $ARFD$s obtained with the same relative number of swaps in the considered real datasets are very different. This is due to the different distributions of the sequential patterns in the datasets. In particular, SIGN, that has values greater than the other datasets, does not contain repeated items in its transactions, and therefore a series of itemsets swaps changes its distribution more than on the other datasets.

6.5.3 Computational Time of the Generation Strategies

In this Section, we compare the execution times to generate random datasets using the three generation strategies we proposed. For each real dataset and for each generation strategy, we reported the average execution time required to generate a single random

Table 6.2 Execution time to generate a random dataset. The table reports: Dataset $\mathcal{D}$: name of the dataset; for each generation strategy, itemsets swapping (IS), dataset permutation (DP), and transactions permutations (TP), Generation Time (ms): the time to generate a single random dataset in ms.

Dataset $\mathcal{D}$	Generation Time (ms)		
	IS	DP	TP
BIBLE	490	12.2	9.09
BIKE	43.0	1.68	1.89
FIFA	412	11.3	8.08
FLIGHT	1.35M	51.6K	5.77K
FLIGHT1	792K	33.9K	3.97K
FLIGHT2	1.07M	18.6K	2.52K
LEVIATHAN	43.7	2.27	2.11
SIGN	5.95	0.49	0.42

dataset, over 10 runs. Given the results of Section 6.5.2, for the itemsets swapping strategy we considered $2m$ itemsets swap operations to generate a single random dataset. Table 6.2 shows the results. It is possible to notice that the two strategies based on permutations, i.e., dataset permutation and transactions permutations, are very efficient, since they required a few milliseconds to generate a random dataset (excluding the FLIGHT datasets). In particular, the transactions permutations resulted the fastest one. The itemsets swapping strategy, instead, resulted at least one order of magnitude slower than the other approaches. Given the results of this Section and of the previous one, the dataset permutation strategy results a better strategy than the itemsets swapping one. Finally, it is clear that it is impractical to execute ProMiSe on massive datasets such as FLIGHT, since the generation of a single random dataset, even with the fastest strategy, required several seconds. Thus, strategies based on MC procedures, which require the generation of a large number of random datasets, must be avoided for such massive datasets.

6.5.4 Results of ProMiSe

In this Section, we report the results of ProMiSe for mining significant sequential patterns in real datasets. We did not report results for the itemsets swapping strategy since they were analogous of the ones obtained with the dataset permutation strategy and the dataset permutation strategy allows to obtain them more efficiently. First, we used the BIKE dataset to evaluate the impact of the number P of permuted datasets

Table 6.3 Results of ProMiSe for BIKE. The table reports: Dataset $\mathcal{D}$: name of the dataset; Strategy: generation strategy for the random datasets, itemsets swapping (IS) or dataset permutation (DP); P: number of random datasets for the WY method; M: number of random datasets for the MC procedure; $|FSP|$: number of frequent sequential patterns in $\mathcal{D}$; $|SSP|$: number of significant sequential patterns reported by ProMiSe.

Dataset $\mathcal{D}$	Strategy	P	M	$\lvert FSP\rvert$	$\lvert SSP\rvert$
BIKE ($\theta = 0.025$)	DP	100	1K	163	0
		100	10K		38
		100	100K		38
		1K	1K		0
		1K	10K		38
	TP	100	1K	163	0
		100	10K		29
		100	100K		31
		1K	1K		0
		1K	10K		31

for the WY method and of the number M of random datasets for the MC procedure. Table 6.3 reports the obtained results. For $M = 1000$, ProMiSe did not report any significant sequential patterns with both generation strategies since the MC estimates of the p-values are too coarse (the smallest p-value that can be estimated is 1/1001). For the remaining values, ProMiSe with the DP strategy reported the same amount of patterns, 38. Instead, with the TP strategy, ProMiSe reported 29 patterns with $P = 100$ and $M = 10000$, while it reported 31 patterns with the remaining values. Given these results, we fixed $P = 100$ and $M = 10000$ for all the remaining analyses, since they represent a good trade-off between the number of reported patterns and the execution time.

Then, we executed ProMiSe on the remaining datasets. Table 6.4 reports the results. First of all, let us note that using the dataset permutation strategy, ProMiSe reported as significant all the frequent sequential patterns extracted from SIGN: this is due to the absence of repeated items in the transactions of SIGN, which is therefore extremely different from a random dataset. Thus, SIGN is an example of dataset that requires the TP strategy. For all the datasets and for both generation strategies, ProMiSe returned some significant sequential patterns. In addition, the sequential patterns flagged as significant are always less than the frequent sequential patterns, confirming that the frequency by itself is not enough to provide statistical significance. For FIFA, only one frequent sequential pattern was not reported in output using the DP strategy. This

Table 6.4 Results of ProMiSe. See Table 6.3 for the meaning of the reported values.

Dataset $\mathcal{D}$	Strategy	$\|FSP\|$	$\|SSP\|$
BIBLE ($\theta = 0.1$)	DP	174	119
	TP		121
FIFA ($\theta = 0.275$)	DP	182	181
	TP		142
LEVIATHAN ($\theta = 0.15$)	DP	225	85
	TP		98
SIGN ($\theta = 0.4$)	DP	518	518
	TP		457

is due to the fact that such a dataset contains very few transactions with repeated itemsets, therefore, similarly to what we observed for SIGN, the dataset permutation heavily changes the distribution of the frequencies of the patterns. Indeed, FIFA is the second dataset with the highest $ARFD$ (see Figure 6.1). Let us note that for some datasets, the number of significant sequential patterns found with the TP strategy is greater than the one found with the dataset permutation, although the TP strategy cannot detect sequential patterns of length one or sequential patterns composed by a single itemset repeated multiple times. Such results depend on the distribution of the sequential patterns in the real datasets, highlighting that our two generation strategies provide complementary assessments of the significance of the sequential patterns.

Finally, we assessed the false positives guarantees of ProMiSe using pseudo-artificial datasets. Starting from SIGN, we generated 50 random datasets. Let us note that such random datasets did not contain any significant patterns since they have been sampled from the distribution described by the null hypothesis. Then, we executed ProMiSe on such random datasets, and we estimated the FWER as the fraction of runs in which ProMiSe reported at least one significant sequential pattern, which would be a false positive by construction. Let us note that SIGN was the ideal candidate for such an experiment, since ProMiSe mined many significant sequential patterns from it. With the DP strategy, ProMiSe obtained an estimated FWER of 0%, while it obtained 4% with the TP strategy. Let us note that in the runs in which it reported some false positives, it reported exactly one false positive. These results show that the false positives guarantees of ProMiSe are even better than the theoretical ones, which are $\leq 5\%$ using $\alpha = 0.05$, and that $P = 100$ and $M = 10000$ are enough to obtain such guarantees.

Table 6.5 Results of I-ProMiSe. See Table 6.3 for the meaning of the reported values.

Dataset $\mathcal{D}$	Strategy	$\lvert FSP \rvert$	$\lvert SSP \rvert$
BIBLE ($\theta = 0.1$)	DP	174	69
	TP		53
BIKE ($\theta = 0.025$)	DP	163	0
	TP		0
FIFA ($\theta = 0.275$)	DP	182	142
	TP		138
FLIGHT ($\theta = 0.001$)	DP	566	177
	TP		68
FLIGHT1 ($\theta = 0.001$)	DP	652	189
	TP		33
FLIGHT2 ($\theta = 0.001$)	DP	501	152
	TP		20
LEVIATHAN ($\theta = 0.15$)	DP	225	0
	TP		0
SIGN ($\theta = 0.4$)	DP	518	88
	TP		34

Overall, these results show that ProMiSe is a valid strategy to mine significant sequential patterns while correctly controlling the FWER.

6.5.5 Results of I-ProMiSe

In this Section, we report the results of I-ProMiSe for mining significant sequential patterns in real datasets, in particular from the massive three FLIGHT datasets. Table 6.5 reports the obtained results. Let us remember that I-ProMiSe requires the generation of a single random dataset and thus it is incredible faster than the original version of ProMiSe, with the drawback of a lower statistical power. Indeed, I-ProMiSe did not report any significant sequential patterns from BIKE and LEVIATHAN, while ProMiSe did. In addition, also in all the other datasets, I-ProMiSe reported less significant sequential patterns. However, it reported some significant sequential patterns from all the three FLIGHT datasets, with both generation strategies, while ProMiSe would not report any significant sequential pattern in feasible time. In particular, I-ProMiSe required about 30 minutes to mine FLIGHT, the biggest dataset, less than a tenth of the time that ProMiSe required to mine BIBLE using 64 cores.

Overall, these results show that I-ProMiSe is a valid alternative to ProMiSe for mining statistically significant sequential patterns from massive datasets.

6.5.6 Analysis of BIKE with ProMiSe

In this Section, we provide an analysis of some significant sequential patterns returned by ProMiSe from BIKE. We obtained BIKE by downloading the Los Angeles Metro Bike Share Trip Data and performing the following pre-processing steps in order to create a sequential dataset. The Los Angeles Metro Bike Share Trip Data contains a series of trips performed in Los Angeles using the bike sharing service. For each trip, the following information is available, among others: the starting station, the ending station, a unique identifier of the bike, the starting time, the ending time. We collected all the trips made by each bike and we sorted them using the temporal information available. In particular, for each bike, we collected the ordered sequence of station where the bike has been. The items in our dataset are the ID of the stations. Each transaction represents the ordered list of stations in which a certain bike has been. The first station in each transaction is the starting station of the first trip available for that bike. The second one is the ending station of that trip, which becomes the starting station for the following trip and so on. If for any reason one series of stations contained gaps, i.e., the end station of a trip did not correspond to the starting station of the following trip, we split the sequence where the gap happened, creating two transactions.

To mine significant sequential patterns from BIKE, we used ProMiSe with $\theta = 0.025$, $P = 100$, and $M = 10000$. Figure 6.2 shows the 4 most significant sequential patterns (i.e., with the smallest p-values) found with the DP strategy and the 4 most significant ones found with the transactions permutations. Two patterns have been found only with the DP strategy, three have been found only with the TP strategy, while the remaining three have been found with both. Let us note that the two patterns found only with the DP strategy cannot be mined using the TP strategy, since they are composed by the same itemset twice, and thus they always have the same frequency in random datasets generated with transactions permutations. An example of such patterns is the sequential pattern starting from the Union Station West Portal and ending in the same bike station. Such a bike station is located near the Union Station, the main train station of Los Angeles. The other is a sequential pattern starting from 7th & Flower and ending again in the same station, which is located near the Metro Center station in the financial district of Los Angeles. We investigated whether the two itemsets that composed these sequential patterns are consecutive in the transactions in which they appear, indicating that they correspond to single trips, or if they are a combination of multiple trips. We

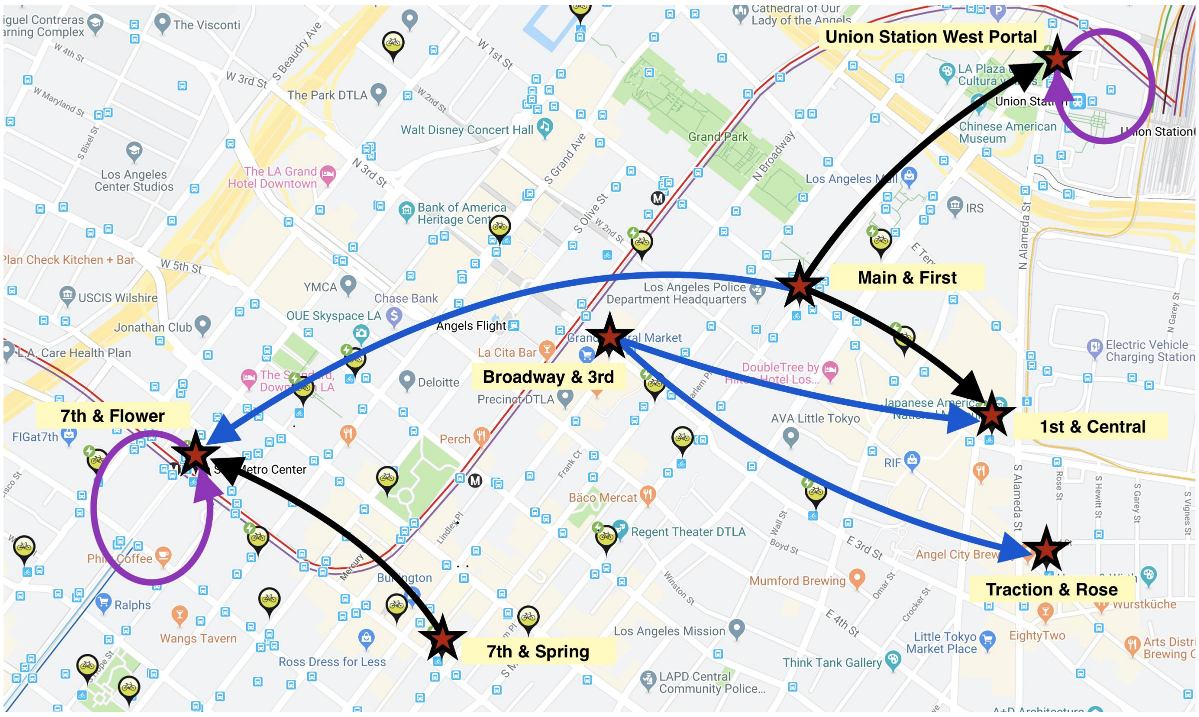

Fig. 6.2 Map of the Los Angeles Metro Bike stations with some significant sequential patterns found with ProMiSe. The red stars are the bike stations involved, with the respective names. Each arrow is a significant sequential pattern: black arrows have been found with both generation strategies; blue arrows have been found only with the TP strategy; purple arrows have been found only with the DP strategy.

considered that two itemsets of a sequential pattern are consecutive in a transaction if there was at least one instance of such a pattern with itemsets in consecutive positions in the transaction. For both the sequential patterns that start and end in the same station, the percentage of single trips is about 25%. The only significant sequential pattern that has a lower percentage is the one that starts from Main & First and that ends in 7th & Flower. This is probably due to the large distance between the two stations. Instead, the sequential pattern with the highest percentage of single trips is the one that starts from Main & First and that ends in Union Station West Portal. This sequential pattern probably catches the flow of people that ride to the Union Station.

Chapter 7

Mining Statistically Significant Paths in Time Series Data from an Unknown Network

In this Chapter, we consider the problem of mining statistically significant paths in time series data from an unknown network. Let us remember that in the previous Chapter, we studied the problem of mining statistically significant sequential patterns and we defined efficient strategies to sample datasets from the distribution described by the null hypothesis, which were at the core of our approach. In this Chapter, instead, we consider the mining of paths in time series data generated by an unknown network, which is another type of sequential data. In such a scenario, the definition of efficient strategies to sample datasets from the distribution described by the null hypothesis is even more challenging, since such time series are constrained by the structure of a network. To solve this difficult task, we proposed two strategies to generate random data from our generative null model using random walks, which allow us to employ the statistical hypothesis testing framework and techniques based on permutation testing, analogous of the ones used in Chapter 6, to mine significant paths. Part of the contributions described in this Chapter appear in [85], while an extended version is currently under review in the journal Knowledge and Information Systems, invited among the best papers accepted at IEEE ICDM'21.

Caspita: italian exclamation indicating surprise,
e.g., "Caspita! Such significant paths are really surprising."

7.1 Introduction

Time series data mining [32, 70, 17] is a fundamental data mining task that covers a wide range of real-life problems in various fields of research such as economic forecasting, telecommunications, intrusion detection, and gene expression analysis. Even if the common purpose is to extract meaningful knowledge from the data, many different problems and approaches have been proposed over the years, ranging from anomaly detection [93, 34] to motif discovery [44], from clustering [33] to classification [92]. However, in many real life scenarios, time series data are generated by networks, and thus are sequences of vertexes representing paths constrained by the structures and the distributions that define such networks. Very often, one has access to a collection of time series but does not know the distribution on the network that generated them, or the structure of such a network. As an example, consider a survey on the paths traveled by people with the underground service of a given city. In such a scenario, one has a dataset that represents a limited number of paths from a network, defined by the underground structure, but does not know the distribution defined by the entire population that uses such a service.

In this Chapter, we study the problem of mining statistically significant paths from an unknown network. We assume to have a time series dataset, defined as a collection of time series, and that such time series are paths generated from an unknown network. Each time series is then a sequence of vertexes of the network. In such a scenario, we are interested in mining unexpected paths from the dataset. Standard techniques usually use the frequency or the number of occurrences as extraction criteria, with the aim of finding interesting paths, but, in many real applications, such metrics are not enough to find paths that provide useful knowledge. For example, paths that appear only few times in a dataset may be over-represented if we consider the distribution of the network underlying the data, or vice-versa, paths that appear a lot of times may be under-represented. Thus, techniques based on such metrics may led to several spurious discoveries. In addition, since we do not know the network underlying the data, we can not directly find over- or under-represented paths.

We then introduce caSPiTa, an algorithm to find statistically significant paths over- (or under-) represented from time series data considering a generative null model based on meaningful characteristics of the observed dataset, while providing guarantees in terms of the false positives employing the Westfall-Young (WY) method. Our generative null model is based on the observed number of occurrences of paths of a given length, with the idea that such paths represent well-known substructures of the underlying network. In the simplest case, they are paths of length one, that are edges of the underlying network. Then, such a null model is used to test the significance of paths of a given, higher, length, which are the paths of interest mined from the observed dataset. The intuition is to create a generative null model that is able to explain the number of occurrences of shorter paths, and to check whether such a generative null model is able to also explain the observed number of occurrences of the paths of interest. Otherwise, such paths can be considered significant, in the sense that they appear more (or less) times than expected under such a generative null model. Let us consider, as an example, a network composed by the web-pages of a website, and suppose that we want to find sequences of web-pages visited more (or less) than expected w.r.t. the underlying distribution of the network, defined by the navigation of the users on the website. Given the application or the structure of the network, there may be some well-know substructures, defined as short sequences of web-pages, that are traversed by the users that visit such a website with a particular distribution. Again, let us note that in the simplest case, the substructures are paths of length one, that represent a direct link between two web-pages. Thus, in such a scenario, one may be interested in finding if such substructures also explain the number of observed occurrences of longer paths, or if such longer paths are significant due to some external factors causing their number of occurrences.

7.1.1 Our Contributions

In this Chapter, we introduce the problem of mining *statistically significant paths* in time series data from an unknown network. In this regard, our contributions are:

- We introduce the problem of mining statistically significant paths in time series data from an unknown network, defining a generative null model based on meaningful characteristics of the observed dataset.
- We introduce caSPiTa, an algorithm to mine statistically significant paths (over- or under-represented) from a time series dataset, while providing guarantees on the probability of reporting at least one false positive, i.e., the FWER. We also discuss an extension of caSPiTa to simultaneously mine both types of paths.
- We introduce *g*-caSPiTa, a variant of caSPiTa to mine statistically significant paths (over- or under-represented) while providing guarantees on the *generalized*

FWER, which allows to increase the statistical power of the algorithm, with the drawback of tolerating the presence of a few false positives.

- We introduce an alternative interesting scenario in which caSPiTa can be applied, which consists in mining paths that are significant w.r.t. a null model based on data from a different dataset.

- We perform an extensive suite of experiments that demonstrates that caSPiTa is able to efficiently mine statistically significant paths in real datasets while providing guarantees on the false positives.

Let us note that throughout the Chapter, unless otherwise noted, we only describe the scenario in which one is interested in mining statistically significant paths that occur more times than expected under the null hypothesis (*over-represented paths*), for clarity of presentation. However, all the reasoning are still valid to mine paths that occur less times than expected (*under-represented paths*). In particular, we discuss the mining of under-represented paths in Section 7.3.5 and our open source implementation of caSPiTa mines over- and under-represented paths. In addition, results for both scenarios are shown in the experimental evaluation.

7.1.2 Related Work

In significant pattern mining, the dataset is seen as a collection of samples from an unknown distribution, and one is interested in finding patterns significantly deviating from an assumed *null hypothesis*, i.e., *distribution*. Many variants and algorithms have been proposed for the problem. We point interested reader to the survey [26], and recent works that employ permutation testing [45, 60]. Even if our work falls within the framework of significant pattern mining, such approaches are orthogonal to our work, which focuses on finding significant paths, i.e., patterns, from time series that are constrained by a network structure.

Many works have been proposed to detect anomalies in sequential data [93, 23], employing several definition of anomalies, and considering different types of patterns. For example, [34] defines a pattern as *surprising* if its frequency differs substantially from that expected by chance, given some previously seen data. Lemmerich et al. [42], instead, considers the mining of subgroups, defined by subsets of attributes, that exhibit exceptional transition behavior, i.e., induce different transition models compared to the ones of the attributes that describe the entire dataset. Although our approach adopts a definition of significant pattern based on how the number of its occurrences differs from the one expected under an appropriate model, similarly to other works, we consider the setting in which the data represent paths from a weighted and directed graph, which results in a different problem. In fact, this aspect makes our work closer to the task of detecting anomalies in temporal graph [53, 5], i.e., graphs that evolve over time. However, even if our work considers data generated by a network and aims to find paths whose number of occurrences is significant w.r.t. the network's distribution, we consider the scenario in which we do not know the network, and we have only access to a sample.

The only work that considers the problem of finding anomalous paths in time series data from an unknown network is [40]. In this work, the authors propose an algorithm, HYPA, to find anomalous length k paths using a null model based on length $k-1$ paths. In particular, they aim to find length k paths whose number of occurrences in a dataset is anomalous w.r.t. a null model based on the number of occurrences of length $k-1$ paths in the same dataset. Reducing the difficult problem of detecting anomalous length k paths to the easier problem of detecting anomalous edges in a k-th order De Bruijn graph, they describe a strategy based on the hypergeometric distribution to compute a score for each length k path, where the score describes the level of anomaly of such a path. Even if our approach is inspired by [40], our work differs from it in many key aspects. First of all, we aim to find length k paths whose number of occurrences in a dataset is significant w.r.t. a null model based on the number of occurrences of length h paths, with $h \in \{1, \ldots, k-1\}$ provided in input by the user, and not only with $h = k-1$ as in [40]. In such a direction, it is not clear if HYPA can be modified to consider a more general length $h \in \{1, \ldots, k-1\}$. Finally, while our approach employs the WY method to correct for multiple hypothesis testing providing guarantees in terms of false positives, [40] uses fixed thresholds to define interesting patterns, which does not provide any guarantee.

To the best of our knowledge, our work is the first approach that employs the statistical hypothesis testing framework to mine paths, i.e., patterns, from time series constrained by the structure and the distribution of an unknown network, while providing rigorous guarantees on the probability of reporting at least one false positive, i.e., the FWER.

7.1.3 Organization of the Chapter

The rest of the Chapter is structured as follows. Section 7.2 contains the definitions and concepts used throughout this Chapter. Section 7.3 describes our algorithm caSPiTa, besides all related concepts, and the discussion of possible extensions of our approach. Section 7.4 describes an alternative scenario in which caSPiTa can be applied, which considers two datasets. Section 7.5 reports the results of an extensive suite of experiments performed to evaluate the effectiveness of caSPiTa on real and pseudo-artificial datasets.

7.2 Preliminaries

We now provide the definitions and concepts used in the Chapter. First, in Section 7.2.1, we describe the task of mining paths in time series data from a network. Then, in Section 7.2.2, we define, similarly to [74], the concept of k-th order De Bruijn graph used to define our generative null model. Finally, in Section 7.2.3, we refresh concepts of hypothesis and multiple hypothesis testing, and apply it for significant path mining.

7.2.1 Mining Paths in Time Series Data from a Network

Let us define a *network* $N = (G, \omega)$ as a *directed graph* $G = (V, E)$ and a *weight function* $\omega : E \to [0, 1]$. $V = \{v_1, v_2, \dots, v_{|V|}\}$ is the *vertices set*, where each $v \in V$ is called *vertex*, and $E = \{(u, v) : u, v \in V\}$ is the *edges set*, where each (u, v) is an *ordered* pair of vertices, called *edge*. An edge (u, v) is an *incoming edge* of the vertex v and an *outgoing edge* of the vertex u. Denoting with $(u, :)$ an outgoing edge of u, for each vertex $u \in V$, we have

$$\sum_{(u,:) \in E} \omega((u, :)) = 1,$$

that is, the weights of the edges from u represent a probability distribution. Fig. 7.1 (left) shows an example of network.

A *path* $w = \{v_{i_0}, v_{i_1}, \dots, v_{i_{|w|}}\}$ of length $|w|$ on the network N is an ordered sequence of $|w| + 1$ vertices such that $(v_{i_j}, v_{i_{j+1}}) \in E\ \forall j \in \{0, \dots, |w| - 1\}$. Let us note that a vertex $v \in V$ is a path $w = \{v\}$ of length $|w| = 0$. A path $w = \{w_0, w_1, \dots, w_{|w|}\}$ *occurs* in a path $q = \{q_0, q_1, \dots, q_{|q|}\}$ starting from position $s \in \{0, \dots, |q| - |w|\}$, denoted by $w \subset q^{(s)}$, if and only if $w_0 = q_s, w_1 = q_{s+1}, \dots, w_{|w|} = q_{s+|w|}$. We say that the path w is a *sub-path* of the path q. The number of *occurrences* $Occ_q(w)$ of w in q is the number of times that w occurs in q, that is,

$$Occ_q(w) = |\{s \in \{0, \dots, |q| - |w|\} : w \subset q^{(s)}\}|.$$

A *time series dataset* $\mathcal{D} = \{\tau_1, \tau_2, \dots, \tau_{|\mathcal{D}|}\}$ from a network N is a bag of $|\mathcal{D}|$ *transactions*, which are paths on N. Given a path w on N, the number of occurrences $Occ_{\mathcal{D}}(w)$

of w in $\mathcal{D}$ is the sum of the number of occurrences $Occ_\tau(w)$ of w in τ, $\forall\, \tau \in \mathcal{D}$, that is,

$$Occ_{\mathcal{D}}(w) = \sum_{\tau\in\mathcal{D}} Occ_\tau(w).$$

Given a positive integer ℓ, the task of *mining paths* of length ℓ from a time series dataset $\mathcal{D}$ from a network N is the task of mining the set $\mathcal{W}_{\mathcal{D}}(\ell)$ of all paths of length ℓ that occur at least once in $\mathcal{D}$ and the number of their occurrences, that is,

$$\mathcal{W}_{\mathcal{D}}(\ell) = \{(w, Occ_{\mathcal{D}}(w)) : |w| = \ell \wedge Occ_{\mathcal{D}}(w) > 0\}.$$

With an abuse of notation, in the following we use $w \in \mathcal{W}_{\mathcal{D}}(\ell)$ to indicate that $\exists (w, Occ_{\mathcal{D}}(w)) \in \mathcal{W}_{\mathcal{D}}(\ell)$.

7.2.2 k-th Order De Bruijn Graph

Given a directed graph $G = (V, E)$ and an integer $k > 0$, the *k-th order De Bruijn graph* $G^k = (V^k, E^k)$ of G is a directed graph where each vertex $v^k \in V^k$ is a path of length $k-1$ on G, i.e., $v^k = \{v_{i_0}, v_{i_1}, \ldots, v_{i_{k-1}}\}$, and an ordered pair (v^k, u^k), with $v^k = \{v_{i_0}, v_{i_1}, \ldots, v_{i_{k-1}}\}$, $u^k = \{u_{j_0}, u_{j_1}, \ldots, u_{j_{k-1}}\} \in V^k$, it is an edge of G^k if and only if $v_{i_t} = u_{j_{t-1}}$ $\forall t \in \{1, \ldots, k-1\}$. Thus, each edge $(v^k, u^k) \in E^k$ is a path of length k on G, since $(v^k, u^k) = \{v_{i_0}, v_{i_1} = u_{j_0}, v_{i_2} = u_{j_1}, \ldots, v_{i_{k-1}} = u_{j_{k-2}}, u_{j_{k-1}}\}$. Let us note that G itself is a 1-st order De Bruijn graph of G. Fig. 7.1 (right) shows an example of k-th order De Bruijn graph.

Example 10. *Let us consider as an example the network $N = (G, \omega)$ (left) and the 2-nd order De Bruijn graph $G^2 = (V^2, E^2)$ of G (right), both shown in Fig. 7.1. The network $N = (G, \omega)$ is composed by the directed graph $G = (V, E)$, with $V = \{A, B, C, D\}$ and $E = \{(A, C), (A, D), (B, A), (B, D), (C, B)\}$, and by the weight function ω, such that $\omega((A, C)) = 0.3$, $\omega((A, D)) = 0.7$, $\omega((B, A)) = 0.8$, $\omega((B, D)) = 0.2$, and $\omega((C, B)) = 1.0$. The paths $w = CBAC$ and $q = BAD$ are example of paths on N, respectively of length $|w| = 3$ and $|q| = 2$. The 2-nd order De Bruijn graph $G^2 = (V^2, E^2)$ of G is composed by $V^2 = \{AC, AD, BA, BD, CB\}$, where each vertex $v^2 \in V^2$ represents a path of length 1 on G, and by $E^2 = \{(AC, CB), (BA, AC), (BA, AD), (CB, BA), (CB, BD)\}$, where each edge $(v^2, u^2) \in E^2$ represents a path of length 2 on G, i.e., ACB, BAC, BAD, CBA, and CBD.*

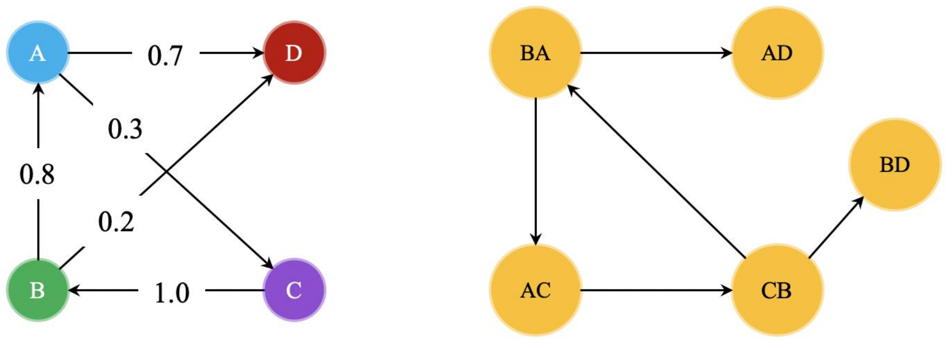

Fig. 7.1 Example of network and of De Bruijn graph. It shows the network $N = (G, \omega)$ (left) and the 2-nd order De Bruijn graph $G^2 = (V^2, E^2)$ of G (right).

7.2.3 Multiple Hypothesis Testing for Paths

The task of mining statistically significant paths requires to identify paths whose number of occurrences in a dataset $\mathcal{D}$ is *significant*, or unexpected, w.r.t. the distribution of the weight function of the network that generated such data. To assess the significance of a path, the framework of statistical hypothesis testing (see Section 2.5.1) can be employed. For each path w, let H_w be the null hypothesis that the number of occurrences $Occ_{\mathcal{D}}(w)$ of w on $\mathcal{D}$ well conforms to the number of its occurrences in *random* time series data generated from the network $N = (G, \omega)$. We define a *random dataset* $\tilde{\mathcal{D}}$ as a collection of random data which contains the same number of paths of interest, i.e., length k paths, of the original dataset $\mathcal{D}$ and that is generated from the graph G in accordance with the weight function ω. That is, a path from a vertex $u \in V$ continues with vertex $v \in V$ with probability $\omega(u, v)$. In addition, the starting vertices (of full transactions or of single paths) observed in the original dataset $\mathcal{D}$ are preserved in the random dataset $\tilde{\mathcal{D}}$.

Let us remember that under the null hypothesis, the number of occurrences of w is described by a r.v. X_w, and in order to assess the significance of w, a p-value p_w is commonly computed. In our scenario, the p-value p_w of w is the probability of observing a number of occurrences, under the null hypothesis, at least as large as the number of occurrences $Occ_{\mathcal{D}}(w)$ of w in $\mathcal{D}$, that is,

$$p_w = \Pr\left[X_w \geq Occ_{\mathcal{D}}(w) | H_w\right].$$

Let us note that for our null hypotheses there is not a closed form for X_w and thus the p-values can not be computed analytically. However, they can be estimated by a

Monte Carlo (MC) procedure (see Section 2.5.1) as

$$p_w = \frac{1}{M+1}\left(1 + \sum_{i=1}^{M} \mathbb{1}\left[Occ_{\tilde{\mathcal{D}}_i}(w) \geq Occ_{\mathcal{D}}(w)\right]\right), \tag{7.1}$$

where $\tilde{\mathcal{D}}_i$, with $i \in \{1, \dots, M\}$, are M random time series datasets generated in accordance with the distribution described by the null hypothesis.

As already discussed, the statistical hypothesis testing framework is commonly used to provide guarantees on the false discoveries, i.e., paths flagged as significant while they are not. When a single path w is tested for significance, flagging w as significant when $p_w \leq \alpha$, where $\alpha \in (0, 1)$ is the *significance threshold*, guarantees that the probability that w corresponds to a false discovery $\leq \alpha$.

The situation is completely different when several paths are tested simultaneously since the expected number of false discoveries, for a fixed α, linearly grows with the size of the set of tested hypotheses, posing a severe *multiple hypothesis correction problem* [58]. To solve this issue, a common approach is to identify a *corrected significance threshold* $\delta \in (0, 1)$ that provides guarantees on the *Family-Wise Error Rate (FWER)*, defined as the probability of reporting at least one false positive.

The Westfall-Young (WY) method [95] is a multiple hypothesis testing procedure based on permutation testing that results in high statistical power. As we discussed in Section 2.5.1, it directly estimates the joint distribution of null hypotheses using P datasets obtained from the distribution described by the null hypothesis. For every random dataset $\tilde{\mathcal{D}}_i$, with $i \in \{1, \dots, P\}$, it computes the minimum p-value $p_{min}^{(i)}$ over all paths of interest in $\tilde{\mathcal{D}}_i$. Finally, it estimates the corrected significance threshold δ^* as the α-quantile of the P minimum p-values, that is,

$$\delta^* = \max\left\{\delta : \sum_{i=1}^{P} \mathbb{1}\left[p_{min}^{(i)} \leq \delta\right] \leq \alpha P\right\}, \tag{7.2}$$

with $\alpha \in (0, 1)$ the *FWER threshold*.

7.3 caSPiTa: Mining Statistically Significant Paths

In this Section, we describe our method caSPiTa, mining statistiCAlly Significant Paths In Time series dAta, to mine *statistically significant paths* in time series data generated by a network, while controlling the probability of having at least one false discovery, i.e., the FWER. Given a time series dataset $\mathcal{D}$ from an *unknown* network N, we aim to mine statistically significant paths, which are paths that have a number of occurrences on $\mathcal{D}$ that is surprising, i.e., higher than the expected number of their occurrences under the null hypothesis. In particular, given two natural values $k, h \in \mathbb{N}^+$, with $k > h$, we aim to mine length k paths from $\mathcal{D}$ whose number of occurrences are not due to the number of occurrences of length h paths observed in $\mathcal{D}$, with the idea that such paths of length h represent some well-know substructures in the underlying and unknown network.

The idea behind caSPiTa is the following. First, we mine all the paths $\mathcal{W}_{\mathcal{D}}(k)$ of length k from the time series dataset $\mathcal{D}$. Since we do not know the network N from which $\mathcal{D}$ has been generated, we can not directly infer the statistical significance of such paths w.r.t. N, and thus we need to construct a new network, i.e., a generative null model, from the dataset $\mathcal{D}$. Such a generative null model is then used to generate random time series datasets in order to estimate the p-values and to compute the corrected significance threshold δ^* using the WY method. We now describe the generative null model employed by caSPiTa.

7.3.1 Generative Null Model

In this Chapter, we aim to find length k paths whose number of occurrences in $\mathcal{D}$ are not due to the number of occurrences of shorter length h paths in $\mathcal{D}$. Thus, we construct a generative null model in accordance with the number of occurrences of the paths of length h in $\mathcal{D}$, and then we test the significance of the paths of length k using such a model. Given a time series dataset $\mathcal{D}$, generated by an unknown network $N = (G, \omega)$, and $h \in \mathbb{N}^+$, we define the *h-th order generative model* $N^h(\mathcal{D})$ of the time series dataset $\mathcal{D}$ as a network $N^h(\mathcal{D}) = (G^h, \omega^h)$, where G^h is the h-th order De Bruijn graph of G (based on $\mathcal{D}$, since the entire structure of G is unknown), and ω^h is a weight function. The h-th order De Bruijn graph $G^h = (V^h, E^h)$ is composed as follows: $V^h = \{w \in \mathcal{W}_{\mathcal{D}}(h-1)\}$, while E^h is constructed as defined in Definition 7.2.2. Thus, each vertex $v^h \in V^h$ is a path of length $h-1$ in $\mathcal{D}$ (and thus on G), while each edge $(u^h, v^h) \in E^h$ represents a path of length h in $\mathcal{D}$ (and thus on G). With an abuse of notation, in the following we use (u^h, v^h) to indicate both the edge in G^h and the corresponding path on G. Finally, the weight function ω^h is defined as follows: $\forall (u^h, v^h) \in E^h$,

$$\omega^h\left((u^h, v^h)\right) = \frac{Occ_{\mathcal{D}}\left((u^h, v^h)\right)}{\sum_{(u^h,:)\in E^h} Occ_{\mathcal{D}}\left((u^h, :)\right)}.$$

Let us note that the weight function ω^h of the h-th order generative null model $N^h(\mathcal{D})$ is defined using the observed number of occurrences of length h paths, and that each

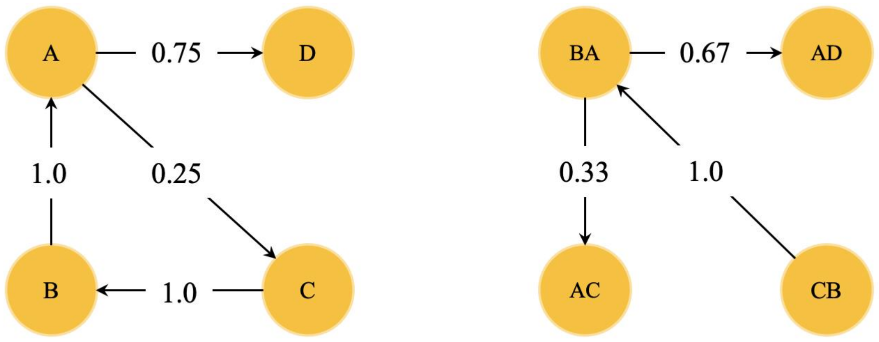

Fig. 7.2 Example of generative null models. It shows $N^1(\mathcal{D})$ (left) and $N^2(\mathcal{D})$ (right), respectively the 1-st and the 2-nd order generative null model of $\mathcal{D} = \{\tau_1 = BAD, \tau_2 = CBAC, \tau_3 = CBAD, \tau_4 = AD\}$.

edge of $N^h(\mathcal{D})$ represents a path of length h. Thus, $N^h(\mathcal{D})$ correctly represents the distribution of the number of occurrences of the paths of length h in the dataset $\mathcal{D}$. An example of generative null models is shown in Fig. 7.2.

Example 11. *Let us consider the following dataset $\mathcal{D} = \{\tau_1, \tau_2, \tau_3, \tau_4\}$, which is a possible time series dataset from the network N shown in Fig. 7.1 (left), as an example:*

$$\tau_1 = BAD$$
$$\tau_2 = CBAC$$
$$\tau_3 = CBAD$$
$$\tau_4 = AD.$$

Fig. 7.3 shows $N^1(\mathcal{D})$ (left) and $N^2(\mathcal{D})$ (right), respectively the 1-st and the 2-nd order generative null model of $\mathcal{D}$. Let us note that the two generative models represent different probability distributions for the paths of length $> h$, e.g., the path $w = BAD$ has a probability of 0.75 of being generated in $N^1(\mathcal{D})$ while it has a probability of 0.67 in $N^2(\mathcal{D})$. In addition, let us note that they are based on the dataset $\mathcal{D}$ and not on the network N that generated $\mathcal{D}$. Indeed, they have some missing edges w.r.t. the network N (that is a 1-st order De Bruijn graph of itself) and its 2-nd order De Bruijn graph G^2, respectively, both shown in Fig. 7.1.

As explained in Section 7.2.3, to compute the p-values p_w of the paths $w \in \mathcal{W}_{\mathcal{D}}(k)$ and to estimate the corrected significance threshold δ^*, we require random data generated from the generative null model. In the following two Sections, we introduce two different strategies to generate random datasets $\tilde{\mathcal{D}}$ that contain the same total number T of paths of length k of the original dataset $\mathcal{D}$, that is,

$$T = \sum_{w \in \mathcal{W}_{\mathcal{D}}(k)} Occ_{\mathcal{D}}(w) = \sum_{w \in \mathcal{W}_{\tilde{\mathcal{D}}(k)}} Occ_{\tilde{\mathcal{D}}}(w). \tag{7.3}$$

First, we describe the *transactions oriented generation* (TOG) strategy, a natural way to generate random datasets performing a series of random walks that generate random transactions with characteristics similar to the ones of the transactions in $\mathcal{D}$. To overcome some issues of this strategy when it is applied to large generative null model, we then introduce the *paths oriented generation* (POG) strategy, an alternative approach that directly generates random length k paths. For this second strategy, we also introduce an approximation based on the binomial distribution that allows to estimate the p-values avoiding expensive MC procedures.

7.3.2 Transactions Oriented Generation (TOG) Strategy

In this Section, we explain how to generate random datasets $\tilde{\mathcal{D}}$ from the generative null model $N^h(\mathcal{D})$ defined above using the TOG strategy. The idea is to perform a series of random walks that generate random transactions $\tilde{\tau}$ with characteristics similar to the ones of the transactions $\tau \in \mathcal{D}$. Characteristics that are natural to consider and that we want to preserve are:

- the dataset $\tilde{\mathcal{D}}$ has the same number of transactions of $\mathcal{D}$;
- each transaction $\tilde{\tau} \in \tilde{\mathcal{D}}$ has the same length of the corresponding transaction $\tau \in \mathcal{D}$;
- each transaction $\tilde{\tau} \in \tilde{\mathcal{D}}$ starts from the same vertex (of the generative null model) of the corresponding transaction $\tau \in \mathcal{D}$.

Let us note that to preserve such characteristics guarantees to preserve also the total number T of length k paths in the dataset. As a motivation to consider such characteristics, let us consider the case in which the dataset $\mathcal{D}$ contains transactions that represent visits of some users in a website. In such a scenario, we are interested in preserving the web-pages from which the visits start, since they probably are homepages (or web-pages from which the users typically start their navigation). In addition, by preserving the length of the transactions, we preserve the number of web-pages that the users visit in a single navigation on the website.

Let s_τ be a path of length $|s_\tau| = h - 1$ such that $s_\tau \subset \tau^{(0)}$, that is, s_τ is the vertex of $N^h(\mathcal{D})$ from which the transaction τ starts. The TOG strategy is the following. For each $\tau_i \in \mathcal{D}$, we perform a random walk on $N^h(\mathcal{D})$ of $|\tau_i| - (h-1)$ steps, starting from the vertex s_{τ_i}. At each step, the random walk moves from a vertex v^h to a vertex u^h with probability $\omega^h\left((v^h, u^h)\right)$. The path generated from such a random walk is then the transaction $\tilde{\tau}_i \in \tilde{\mathcal{D}}$ that corresponds to the transaction $\tau_i \in \mathcal{D}$, with $|\tau_i| = |\tilde{\tau}_i|$. Performing all the $|\mathcal{D}|$ random walks, we generate the random dataset $\tilde{\mathcal{D}}$. Let us note that a random walk may reach a vertex without outgoing edges before performing the desired number of steps, generating a shorter transaction, and thus not preserving the second characteristic (and neither T). In such a case, we discard the transaction and repeat the random walk until we generate a transaction $\tilde{\tau}_i$ with $|\tilde{\tau}_i| = |\tau_i|$.

The TOG strategy is the most natural way to generate random data from the generative null model. However, when the generative null model is large, as happens in many real applications, the number of paths contained in a dataset is only a small fraction of the gargantuan number of paths that can be generated as sub-paths of such long transactions. Thus, the corrected significance threshold δ^* obtained with the WY method could be very small, resulting in few or even zero reported significant paths. In addition, depending on the structure of the generative null model, to generate such long transactions may be computationally expensive for the high number of transactions that we need to generate and discard before reaching the desired lengths.

7.3.3 Paths Oriented Generation (POG) Strategy

To overcome the issue of the TOG strategy, we now describe an alternative approach to generate random data. In the POG strategy, instead of generating long transactions, we generate single random paths w of length $|w| = k$. In particular, from the generative null model $N^h(\mathcal{D})$ defined above, we generate random datasets $\tilde{\mathcal{D}}$, which are bags of paths of length k, where the number of paths w of length $|w| = k$ that start in each vertex (of the generative null model) is the same in the two datasets $\mathcal{D}$ and $\tilde{\mathcal{D}}$. Let us note that to preserve such a characteristic guarantees to also preserve the total number T of length k paths in the dataset.

Let us remember that s_w is a path of length $|s_w| = h-1$ such that $s_w \subset w^{(0)}$, that is, s_w is the vertex of $N^h(\mathcal{D})$ from which the path w starts, and let $\mathcal{S} = \{s_w : w \in \mathcal{W}_{\mathcal{D}}(k)\}$ be the set of vertices of $N^h(\mathcal{D})$ from which starts at least one path $w \in \mathcal{W}_{\mathcal{D}}(k)$. To generate random paths, for each vertex $s \in \mathcal{S}$, we perform a series of random walks of $k - (h-1)$ steps on $N^h(\mathcal{D})$, until we generate

$$n_s = \sum_{w \in \mathcal{W}_{\mathcal{D}}(k) : s_w = s} Occ_{\mathcal{D}}(w) \tag{7.4}$$

random paths w of length $|w| = k$ that start from such a vertex s. Then, the bag of all the paths of length k generated from all the vertices $s \in \mathcal{S}$ is the random dataset $\tilde{\mathcal{D}}$. Let us note that $|\tilde{\mathcal{D}}| = \sum_{s\in\mathcal{S}} n_s = T$. As explained above, let us remember that at each step the random walk moves from a vertex v^h to a vertex u^h with probability $\omega^h((v^h, u^h))$. Thus, each random walk generates a path of length k or a path of length $< k$ that ends in a vertex without outgoing edges. Since we are interested in paths of length k, we discard all generated paths of length shorter than k.

While the POG strategy overcomes the issue of the TOG strategy explained above reducing the space of paths that can be generated from the generative null model, it still requires expensive MC procedures to estimate the p-values, and such procedures could be computationally prohibitive for large datasets. In the following Section, we introduce a method to approximate the p-values for the POG strategy avoiding the MC procedure.

Binomial Approximation for the p-values

In this Section, we illustrate an approach to approximate the p-values p_w of paths w of length $|w| = k$ when the POG strategy is used to generate random data. First, we compute with which probabilities such paths are generated under the POG strategy. Let us consider a random walk that starts from a vertex $s \in \mathcal{S}$ and that performs $k-(h-1)$ steps on $N^h(\mathcal{D})$, and let $\mathcal{W}_s$ be the set of all paths that can be generated by such a random walk. As explained above, the set $\mathcal{W}_s$ contains paths of length $|w| = k$ and, eventually, paths of length $|w| < k$ that end in a vertex without outgoing edges. Let $RW(w)$ be the set of edges of $N^h(\mathcal{D})$ that the random walk traverses to generate the path $w \in \mathcal{W}_s$. From the definition of random walk, the probability $\Pr(w)$ that the random walk generates $w \in \mathcal{W}_s$ starting from s is

$$\Pr(w) = \prod_{(u^h,v^h)\in RW(w)} \omega^h\left((u^h, v^h)\right).$$

Let us note that $\sum_{w\in\mathcal{W}_s} \Pr(w) = 1$. Let E_k be the event that the random walk starting from s generates a path of length exactly k and let $\mathcal{W}_s^k \subseteq \mathcal{W}_s$ be the set of paths $w \in \mathcal{W}_s$ with $|w| = k$. Since in the POG strategy we discard paths shorter than k which could be generated by the series of random walks, then the probability of generating the path $w \in \mathcal{W}_s^k$ is

$$\Pr(w \mid E_k) = \frac{\Pr(w \cap E_k)}{\Pr(E_k)}, \tag{7.5}$$

where $\Pr(w \cap E_k) = \Pr(w)$ for all $w \in \mathcal{W}_s^k$ and 0 otherwise, and $\Pr(E_k) = \sum_{w \in \mathcal{W}_s^k} \Pr(w)$. Again, let us note that $\sum_{w \in \mathcal{W}_s^k} \Pr(w \mid E_k) = 1$, and that if $\mathcal{W}_s \backslash \mathcal{W}_s^k = \emptyset$, then $\Pr(w \mid E_k) = \Pr(w)$.

Example 12. *Let us consider the* 2*-nd order generative null model* $N^2(\mathcal{D})$ *shown in Figure 7.3 (left), as an example. For* $k = 3$*, starting from the vertex* BA *and performing* $k - (h-1) = 2$ *steps, a random walk can generate the paths of length* 3 $BAAE$*,* $BAAB$ *and* $BACE$*, or can reach the vertex* AD *just after one step, generating the path of length* 2 BAD*. The probabilities of all these paths are:* $\Pr(BAAE) = 0.15$*,* $\Pr(BAAB) = 0.35$*,* $\Pr(BACE) = 0.3$*, and* $\Pr(BAD) = 0.2$*, and then the probability of generating a path of length* 3 *starting from the vertex* BA *is* $\Pr(E_3) = 0.8$*. Thus, the probabilities of the length* 3 *paths under the POG strategy are:* $\Pr(BAAE \mid E_3) = 0.1875$*,* $\Pr(BAAB \mid E_3) = 0.4375$*, and* $\Pr(BACE \mid E_3) = 0.375$*, shown in Figure 7.3 (right).*

Since from a given vertex $s \in \mathcal{S}$, we generate exactly n_s (see Equation 7.4) length k paths, then the number of occurrences $Occ_{\tilde{\mathcal{D}}}(w)$ of a path $w \in \mathcal{W}_s^k$ in the random dataset $\tilde{\mathcal{D}}$ follows a binomial distribution, that is, $Occ_{\tilde{\mathcal{D}}}(w) \sim \text{Bin}(n_s, \Pr(w \mid E_k))$. For a fixed vertex $s \in \mathcal{S}$, this is true for all the paths $w \in \mathcal{W}_s^k$, but the binomial distributions corresponding to these paths are not independent, and thus, the computation of the p-values p_w as

$$p_w = \Pr\left[\text{Bin}(n_s, \Pr(w \mid E_k)) \geq Occ_{\mathcal{D}}(w)\right] \tag{7.6}$$

considers a number of paths that is in *expectation* n_s, and not exactly n_s as for the original POG strategy. (Note that, as a consequence, the *total* number of considered paths of length k is T *in expectation*.) However, in our experimental evaluation, we empirically show that the p-values for the binomial approximation are within one order of magnitude of the corresponding MC p-values, and, thus, that the binomial approximation is a valid approach to approximate the p-values for the POG strategy, avoiding expensive MC procedures.

Let us note that while this approximation does not require the generation of M random datasets $\tilde{\mathcal{D}}$ to estimate the p-values, caSPiTa still requires the generation of P random datasets $\tilde{\mathcal{D}}$ for the WY method. However, the binomial approximation can also be used to approximate the minimum p-value in the P random datasets. Thus, given the observed dataset $\mathcal{D}$, we compute the p-values for all paths $w \in \mathcal{W}_{\mathcal{D}}(k)$ using Equation 7.6. Then, we generate a series of P random datasets $\tilde{\mathcal{D}}$ required by the WY method using the POG strategy. For all the P random datasets $\tilde{\mathcal{D}}$, we compute the minimum p-value over all the paths $w \in \mathcal{W}_{\tilde{\mathcal{D}}}(k)$, where the p-value p_w of w is computed with Equation 7.6 replacing $Occ_{\mathcal{D}}(w)$ with $Occ_{\tilde{\mathcal{D}}}(w)$.

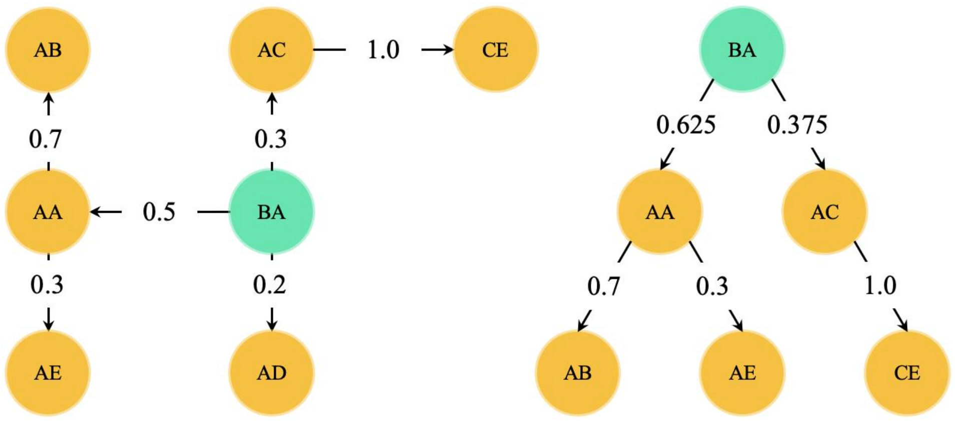

Fig. 7.3 Example of paths oriented generation. It shows the 2-nd order generative null model $N^2(\mathcal{D})$ and the starting vertex BA (left), and the probabilities of all paths of length 3 under the POG strategy (right).

7.3.4 Analysis

In this Section, we describe in detail our algorithm caSPiTa and formally prove its false positives guarantees. Algorithm 13 shows the pseudo-code of caSPiTa. Its inputs are the time series dataset $\mathcal{D}$, the FWER threshold $\alpha \in (0,1)$, the order $h > 0$ of the the generative null model, and the paths length $k > h$. For a given generation strategy, (i.e., TOG or POG), caSPiTa first mines the set $\mathcal{W}_{\mathcal{D}}(k)$, of all paths w of length $|w| = k$ that occur at least once in $\mathcal{D}$ (line 1). Then, it constructs the generative null model $N^h(\mathcal{D})$ (line 2) as explained in Section 7.3.1, and it uses $N^h(\mathcal{D})$ to compute the p-values of the paths $w \in \mathcal{W}_{\mathcal{D}}(k)$ (line 4). The p-values can be computed with a MC procedure using Equation 7.1 (for both generation strategies), and thus generating M random datasets, where M is a parameter set by the user, or with the binomial approximation using Equation 7.6 (for the POG strategy). To compute the corrected significance threshold δ^*, it then employs the WY method, which requires the generation of P random datasets (line 6), where P is a parameter set by the user. For each random dataset $\tilde{\mathcal{D}}_i$, with $i \in \{1, \ldots, P\}$, it mines the set $\mathcal{W}_{\tilde{\mathcal{D}}_i}(k)$ (line 7) and then it computes the minimum p-value $p_{min}^{(i)}$ over all paths $\tilde{w} \in \mathcal{W}_{\tilde{\mathcal{D}}_i}(k)$ (line 9). For the computation of such p-values, the considerations made above are still valid. The corrected significance threshold δ^* is then computed using Equation 7.2 (line 11). If $\delta^* > \alpha$, then we set $\delta^* = \alpha$, corresponding to an uncorrected threshold. Finally, the output is the set of paths $w \in \mathcal{W}_{\mathcal{D}}(k)$ such that $p_w < \delta^*$ (line 12). Theorem 22 proves that the output of caSPiTa has FWER $\leq \alpha$.

Theorem 22. *The output of* caSPiTa *has FWER* $\leq \alpha$.

Algorithm 13: caSPiTa

```
   Data: Time Series Dataset $\mathcal{D}$, FWER Threshold $\alpha \in (0,1)$, Order $h > 0$ of the
         Generative Null Model, Paths Length $k > h$.
   Result: Set $\mathcal{SW}$ with FWER $\leq \alpha$.
1  $\mathcal{W} \leftarrow$ MinePaths($\mathcal{D}, k$);
2  $N^h \leftarrow$ GenerativeNullModel($\mathcal{D}, h$);
3  foreach $w \in \mathcal{W}$ do
4      $p_w \leftarrow$ PValue$\big(N^h, w, Occ_{\mathcal{D}}(w)\big)$;
5  for $i \leftarrow 1$ to $P$ do
6      $\tilde{\mathcal{D}}_i \leftarrow$ RandomDataset($N^h, k, h$);
7      $\mathcal{W}_i \leftarrow$ MinePaths($\tilde{\mathcal{D}}_i, k$);
8      foreach $\tilde{w} \in \mathcal{W}_i$ do
9          $p_{\tilde{w}} \leftarrow$ PValue$\big(N^h, \tilde{w}, Occ_{\tilde{\mathcal{D}}_i}(\tilde{w})\big)$;
10     $p_{min}^{(i)} \leftarrow \min\{p_{\tilde{w}} : \tilde{w} \in \mathcal{W}_i\}$;
11 $\delta^* \leftarrow \min\left\{\alpha, \max\left\{\delta : \sum_{i=1}^{P}\left(\mathbb{1}[p_{\min}^{(i)} \leq \delta]\right) \leq \alpha P\right\}\right\}$;
12 $\mathcal{SW} \leftarrow \{(w, Occ_{\mathcal{D}}(w), p_w) : w \in \mathcal{W} \wedge p_w < \delta^*\}$;
13 return $\mathcal{SW}$;
```

Proof. Let us consider the P random datasets $\tilde{\mathcal{D}}_i$, with $i \in \{1, \dots, P\}$, generated by caSPiTa for the WY method. Let us note that they do not contain any significant paths of length k, since they are generated from the generative null model N^h, and thus from the distribution described by the null hypothesis. Given $\delta \in (0,1)$, the FWER FWER(δ) obtained using δ as significance threshold can be estimated using Equation 2.3. That is, estimated as the fraction, over P, of the number of datasets $\tilde{\mathcal{D}}_i$ that contain at least one path with p-value $\leq \delta$, and thus a path that would be reported as significant while it is not when δ is used as significance threshold. Since caSPiTa uses the corrected significance threshold $\delta^* = \max\{\delta : \text{FWER}(\delta) \leq \alpha\}$, then its output has FWER $\leq \alpha$, which concludes the proof. $\square$

We now provide a brief analysis of the time complexity of caSPiTa. The time complexity t_W to mine $\mathcal{W}_{\mathcal{D}}(k)$ and t_N to construct $N^h(\mathcal{D})$ are $t_W = t_N = \mathcal{O}(D)$, with $D = \sum_{\tau \in \mathcal{D}} |\tau|$, since they can be done with a single scan of the entire dataset. The time complexity t_P^{MC} to estimate the p-values of all paths $w \in \mathcal{W}_{\mathcal{D}}(k)$ using MC procedures is $t_P^{MC} = \mathcal{O}(M \cdot t_{\tilde{\mathcal{D}}} + |\mathcal{W}(\mathcal{D}(k)|)$, where $t_{\tilde{\mathcal{D}}}$ is the time complexity to generate a random dataset $\tilde{\mathcal{D}}$ and M is the number of random datasets to create. Let us note that $t_{\tilde{\mathcal{D}}}$ depends on the generation strategy, and that it is not trivial to bound it since we do not know in advance the number of random walks that we need to generate $\tilde{\mathcal{D}}$. However, our experimental evaluation empirically proves that such random datasets can be generated with feasible computational time. In addition, the MC procedure is well-suited to parallelization: when C cores are used to compute the p-values considering M random datasets, each core computes the p-values on M/C random datasets, and the results are then aggregated at the end. The time complexity t_P^B to compute the p-values of all paths $w \in \mathcal{W}_{\mathcal{D}}(k)$ using the binomial approximation is instead $t_P^B = \mathcal{O}(|\mathcal{W}(\mathcal{D}(k)|)$, but it first requires the computation of the probabilities of Equation 7.5. Such a computation has time complexity $\mathcal{O}(|\mathcal{S}| \cdot R^{MAX})$, where $|\mathcal{S}|$ is the number of starting nodes and R^{MAX} is the maximum, over all $s \in \mathcal{S}$, number of vertices that can be reached in $k-h$ steps on $N^h(\mathcal{D})$, starting from each vertex s. Finally, the time complexity of the WY method is $t_{WY} = \mathcal{O}(P \cdot (t_{\tilde{\mathcal{D}}} + t_p))$, with P the number of random datasets to generate and t_p one of the two time complexity described above to compute the p-values.

As previously stated, while here we consider the mining of over-represented paths, all our reasoning can be easily adapted to the mining of under-represented paths. Details in such a direction are provided in the following Section.

7.3.5 Mining Under-Represented Paths

In this Section, we describe how caSPiTa can be modified to mine under-represented statistically significant paths. Let us remember that in order to assess the significance of a path w, we require to compute a p-value p_w. To mine under-represented paths, the p-value p_w of w is the probability of observing a number of occurrences, under the null hypothesis, at least as small as the number of occurrences $Occ_{\mathcal{D}}(w)$ of w in $\mathcal{D}$, that is,

$$p_w = \Pr\left[X_w \leq Occ_{\mathcal{D}}(w) | H_w\right],$$

where X_w is the random variable which describes the number of occurrences of w under the null hypothesis. As already said for the over-represented paths, since there is not a closed formula for X_w under our null hypothesis, the p-value p_w must be estimated with a MC procedure (for both generation strategies) or with the binomial approximation (for the POG strategy). Using a MC procedure, p_w can be estimated as

$$p_w = \frac{1}{M+1}\left(1 + \sum_{i=1}^{M} \mathbb{1}\left[Occ_{\tilde{\mathcal{D}}_i}(w) \leq Occ_{\mathcal{D}}(w)\right]\right), \tag{7.7}$$

where $\tilde{\mathcal{D}}_i$, with $i \in \{1, \ldots, M\}$, are M random time series datasets generated from the distribution described by the null hypothesis. Instead, using the binomial approximation,

p_w can be estimated as

$$p_w = \Pr\left[\text{Bin}(n_s, \Pr(w \mid E_k)) \leq Occ_{\mathcal{D}}(w)\right], \tag{7.8}$$

where $\Pr(w \mid E_k)$ is the probability of generating w under the POG strategy and n_s is the number of paths to generate from the same starting vertex of w. Thus, Algorithm 13 can be simply modified to mine under-represented paths computing the p-values p_w of the paths $w \in \mathcal{W}$ (line 4) with a MC procedure using Equation 7.7 (for both generation strategies) or with the binomial approximation using Equation 7.8 (for the POG strategy). The same approach must be employed to compute the p-values $p_{\tilde{w}}$ of the paths $\tilde{w} \in \mathcal{W}_i$ (line 10) in the P random datasets for the WY method.

In the case one is interested in mining both types of paths, over- and under-represented, it is possible to speed up the execution of caSPiTa mining both types of paths simultaneously. Indeed, since the only difference in the two versions of caSPiTa is the computation of the p-values, it is possible to perform all the other operations only once and to compute both types of p-values (both for the paths of the original and of the random datasets for the WY method). Thus, we obtain two different significance thresholds, one for the over and one for the under-represented paths, to test the respectively p-values. Let us note that this approach is a valid strategy to speed up the execution avoiding unnecessary re-computations but the false positives guarantees are still valid for the two types of paths separately, i.e., the set of returned over-represented paths has FWER $\leq \alpha$ and the set of returned under-represented paths has FWER $\leq \alpha$, separately. Instead, if one is interested in mining over- and under-represented paths obtaining false positives guarantees for both types of paths simultaneously, i.e., the set of returned over- and under-represented paths has FWER $\leq \alpha$, the following strategy is a possible solution. Given the FWER threshold $\alpha \in (0,1)$, to compute the two corrected significance thresholds considering $\alpha/2$, i.e.,

$$\delta^* \leftarrow \max\left\{\delta : \sum_{i=1}^{P} \left(\mathbb{1}[p_{\min}^{(i)} \leq \delta]\right) \leq \frac{\alpha P}{2}\right\},$$

each with the respectively p-values. Using the union bound, it is easy to prove that the resulting output, consisting of both over- and under-represented paths, has FWER $\leq \alpha$.

7.3.6 Controlling the Generalized FWER

In this Section, we illustrate how caSPiTa can be modified to mine statistically significant paths while controlling the *generalized* FWER [41]. In several real applications, one may be interested in tolerating a small amount of false discoveries in order to increase the power of detecting significant paths, still obtaining guarantees on the false positives. In such cases, methods to discover significant paths while controlling the generalized FWER are preferred to methods controlling the FWER. Given a positive integer g, the generalized FWER g-FWER is defined as the probability of reporting at least g false positives, that is, if FP is the number of false positives, then $g\text{-FWER} = \Pr[\text{FP} \geq g]$. For a given value δ, let g-FWER(δ) be the g-FWER obtained when δ is used as corrected significance threshold, that is, by reporting as significant all paths with p-value $\leq \delta$. The WY method can be used to estimate the FWER FWER(δ) obtained using δ as corrected significance threshold as

$$g\text{-FWER}(\delta) = \frac{1}{P}\sum_{i=1}^{P} \mathbb{1}\left[p_g^{(i)} \leq \delta\right],$$

where $p_g^{(i)}$ is the g-th smallest p-values over all paths of interest in the random dataset $\tilde{\mathcal{D}}_i$. Thus, given a g-FWER threshold $\alpha \in (0,1)$, the corrected significance threshold δ^* is obtained as

$$\delta^* = \max\{\delta : g\text{-FWER}(\delta) \leq \alpha\}.$$

Algorithm 13 can be simply modified to mine statistically significant paths with g-FWER $\leq \alpha$. To obtain such guarantees, it is sufficient to substitute lines 10 and 11, respectively, with

$$p_g^{(i)} \leftarrow g\text{-th } \min\{p_{\tilde{w}} : \tilde{w} \in \mathcal{W}_i\}$$

and

$$\delta^* \leftarrow \max\left\{\delta : \sum_{i=1}^{P}\left(\mathbb{1}[p_g^{(i)} \leq \delta]\right) \leq \alpha P\right\}.$$

Let g-caSPiTa be such a modified version of our algorithm. (Let us note that 1-caSPiTa corresponds to the original caSPiTa.) Theorem 23 proves that the output of g-caSPiTa has g-FWER $\leq \alpha$.

Theorem 23. *The output of g-caSPiTa has g-FWER $\leq \alpha$.*

The proof is analogous to the proof of Theorem 22.

7.4 Mining Statistically Significant Paths from Different Datasets

In this Section, we illustrate another interesting scenario in which our algorithm caSPiTa can be applied. Let us suppose to have two datasets, $\mathcal{D}_1$ and $\mathcal{D}_2$, and that such two datasets are taken from the same network N, but in different circumstances, e.g., in different temporal points, or maybe that they represent data generated from two different populations, e.g., men and women. In such a scenario, one may be interested in finding paths from one of the two datasets that are statistically significant considering the distribution represented by the other dataset. Thus, it is possible to use a slightly modified version of caSPiTa, considering the dataset $\mathcal{D}_1$ to generate the h-th order generative null model $N^h(\mathcal{D}_1)$ and then, to consider the paths mined from the other dataset $\mathcal{W}_{\mathcal{D}_2}(k)$, and to compute their significance using $N^h(\mathcal{D}_1)$. Differently from the scenario described above, in this setting it is also possible to mine statistically significant paths of length k considering the h-th order generative null model with $k = h$.

7.5 Experimental Evaluation

In this Section, we report the results of our experimental evaluation on multiple pseudo-artificial and real datasets to assess the performance of caSPiTa for mining statistically significant paths from an unknown network.

The goals of the evaluation are the following:

- To prove that for small datasets, caSPiTa is able to find statistically significant paths with both generation strategies, i.e., TOG and POG, using the MC procedure, while for larger datasets the binomial approximation is necessary to provide useful results;
- To prove that the binomial approximation is a valid approach to approximate the p-values for the POG strategy;
- Focusing on the POG strategy with the binomial approximation, to prove that caSPiTa and its modified version g-caSPiTa are able to find large sets of statistically significant paths in pseudo-artificial and real large datasets, while avoiding false positives, and compare caSPiTa with HYPA [40];
- To prove that caSPiTa is able to find statistically significant paths in the scenario in which the generative null model is constructed considering data from an other dataset (see Section 7.4).

7.5.1 Implementation, Environment, and Datasets

We implemented caSPiTa in Java. We performed all the experiments on the same machine with 512 GB of RAM and 2 Intel(R) Xeon(R) CPU E5-2698 v3 @ 2.3GHz, using

Table 7.1 Datasets characteristics. The Table reports: $|\mathcal{D}|$: number of transactions; Avg $|\tau|$: average transaction length; Max $|\tau|$: maximum transaction length; for the 1-st generative null model $N^1(\mathcal{D})$, $|V^1|$: number of vertices; $|E^1|$: number of edges.

Dataset $\mathcal{D}$	$\|\mathcal{D}\|$	Avg $\|\tau\|$	Max $\|\tau\|$	$N^1(\mathcal{D})$	
				$\|V^1\|$	$\|E^1\|$
BIKE10	3025	1.54	11	10	76
BIKE20	5080	1.90	21	20	279
BIKE	38651	7.51	232	237	10269
FLIGHT	17447803	1.63	15	455	69234
WIKI	51307	5.76	434	4169	59530

Java 1.8.0_201. To parallelize the MC procedures, we used Apache Spark Java API version 3.1.1. Our open-source implementation of caSPiTa and the code developed for the tests and to generate the datasets are available at https://github.com/VandinLab/CASPITA. In all the experiments, we fixed the FWER threshold to the commonly used value $\alpha = 0.05$. In all the experiments, we mined over- and under-represented paths simultaneously, considering the FWER guarantees separately for the two types of paths. To compare with HYPA [40], we used their implementation available online,[1] considering the "rpy2" version. In the following, we describe the datasets used in the evaluation, and how we generated them. Their characteristics are shown in Table 7.1:

- BIKE: data on the bike sharing service of Los Angeles. Each vertex is a bike station, while each transaction represents the sequence of bike stations that a given bike visits. We considered the 2019 data of the "Los Angeles Metro Bike Share trip data",[2] containing single trips in the format of starting station, ending station, and an unique numerical identifier of the bike, among other information. We collected the temporal ordered sequence of bike stations that each bike visited. Such a sequence is a transaction in our dataset. In the case of data anomalies, i.e., an ending station of a trip does not correspond to the starting station of the following trip, we split the sequence where the gap happens, creating two transactions.

- BIKE10 and BIKE20: smaller versions of the BIKE dataset. From BIKE, we only considered the 10 or 20 vertices, respectively, that occur most frequently times, and collect all the transactions that only contain such vertices.

[1] https://github.com/tlarock/hypa
[2] https://bikeshare.metro.net/about/data/

- FLIGHT: data of the commercial flights in the USA. Each vertex is an airport, while each transaction represents the sequence of airports visited in a single itinerary by a passenger. We considered the 2019 data of the "Origin and Destination Survey: DB1BCoupon".[3] Such data contains single flights in the format of origin and destination airports, an unique numerical identifier of the itinerary that contains the flight, and the sequence number of the flight inside the itinerary. We collected the temporal ordered sequence of airports that each passenger visited in a single itinerary, sorting the airports using the sequence numbers. Such a sequence is a transaction in our dataset.

- WIKI: it contains human navigation paths on Wikipedia, collected through the human-computation game Wikispeedia [94]. Each vertex is a Wikipedia web-page, while each transaction is a sequence of web-pages visited by an user during a game. We considered the data "paths finished",[4] that represent finished games.

7.5.2 Generation Strategies Comparison

In this Section, we compare the results obtained by caSPiTa with the TOG or POG strategies that employ MC procedures, and the POG strategy that uses the binomial approximation, on BIKE10 and BIKE20.

The experiments have been performed with $P = 1000$, $M = 10^5$, $k \in \{2, \dots, 5\}$, and $h \in \{1, \dots, k-1\}$. The results are reported in Table 7.2. For BIKE10, the smallest dataset, the number of significant paths obtained with the TOG and POG strategies with MC procedures differs from at most 1, for all combinations of parameters. The same is true when the POG strategy with the binomial approximation is used. In all the cases, caSPiTa reported at most 3 statistically significant paths, which is not surprising since BIKE10 only contains few distinct paths. For BIKE20, the situation is different. For some combinations of parameters (shown in bold in Table 7.2), caSPiTa with the MC procedures did not report any significant (over-represented) paths, while it reported some paths (from 1 to 8) when the binomial approximation is used. In all such cases, the MC estimates resulted in a corrected threshold $\delta^* = 1/(M+1)$, corresponding to the minimum achievable p-value considering M random datasets. Thus, to be able to mine paths, one has to consider a larger value of M, which is infeasible with larger datasets, or to resort to the binomial approximation. This phenomenon appeared with $k > h - 1$, that is, when a large number of distinct paths can be generated, even for a small dataset

[3] https://www.transtats.bts.gov/Fields.asp?gnoyr_VQ=FLM
[4] https://snap.stanford.edu/data/wikispeedia.html

Table 7.2 caSPiTa results with BIKE10 and BIKE20. The Table reports: k: paths length; h: order of the null model; for each dataset, BIKE10 and BIKE20, $|\mathcal{W}|$: number of distinct paths of length k; T: number of total paths of length k; for each generation strategy, TOG (T), POG (P), and POG with binomial approximation (B), $|\mathcal{SW}|$: number of significant paths reported, over (+) and under (−) represented.

k	h	BIKE10								BIKE20							
		$\lvert\mathcal{W}\rvert$	T	$\lvert\mathcal{SW}^T\rvert$		$\lvert\mathcal{SW}^P\rvert$		$\lvert\mathcal{SW}^B\rvert$		$\lvert\mathcal{W}\rvert$	T	$\lvert\mathcal{SW}^T\rvert$		$\lvert\mathcal{SW}^P\rvert$		$\lvert\mathcal{SW}^B\rvert$	
				+	−	+	−	+	−			+	−	+	−	+	−
2	1	164	1630	3	3	3	3	3	3	978	4553	11	12	9	8	11	8
3	1	163	575	1	2	2	2	2	2	997	2320	0	9	0	7	**8**	7
	2			1	2	1	2	1	2			2	8	1	6	1	5
4	1	104	210	0	2	0	2	1	2	713	1220	0	4	0	5	**4**	5
	2			0	1	1	1	1	1			0	7	0	4	0	4
	3			0	1	0	0	0	0			0	4	0	2	0	2
5	1	57	83	0	2	0	2	0	2	450	661	0	2	0	2	**2**	2
	2			0	1	1	0	1	0			0	3	0	2	**1**	2
	3			0	0	0	0	0	0			0	1	0	1	0	1
	4			0	0	0	0	0	0			0	0	0	1	0	1

such as BIKE20. This emphasizes the issue of the TOG strategy described above, that is, the gargantuan number of paths that must be considered with the generation of long transactions.

We then compared the p-values from the POG strategy obtained with the MC procedure and the binomial approximation. Note that while in the MC procedure the total number of length k paths starting from a vertex is fixed to the value observed in the data, using the binomial approximation such a property holds only in expectation. Thus, the p-values from the two approaches will be different. However, by comparing the p-values[5] for all paths (over- and under-represented) of BIKE10 and BIKE20 with $k \in \{2, \dots, 5\}$ and $h \in \{1, \dots, k-1\}$, and considering $M \in \{10^4, 10^5, 10^6\}$ random datasets for the MC estimates, we observed that the p-values for the binomial approximation are within one order of magnitude of the corresponding MC p-values, and that the difference between binomial p-values and MC p-values is lower than the standard deviation of the MC estimates (obtained from 5 estimates of the MC p-values). Furthermore, the binomial approximation is several orders of magnitude faster than the MC procedure

[5]We only considered p-values $\geq 1/(M+1)$, since lower p-values require larger M to be correctly estimated with the MC procedure.

(few milliseconds against over 40 seconds, considering the maximum execution time for both strategies and using 8 cores to parallelize the MC estimates).

7.5.3 Results for POG Strategy with Binomial Approximation

Since the results of the previous Section demonstrated that the POG strategy with the binomial approximation is necessary to mine statistically significant paths from large datasets, and that the p-values for the binomial approximation are within one order of magnitude of the corresponding MC p-values, in this Section we focus on such a version of caSPiTa. First, we investigated the false positives guarantees of caSPiTa on pseudo-artificial datasets, also performing a comparison with HYPA, and then we executed it on real datasets.

False Positives Guarantees

In this Section, we report the results of our experimental evaluation to assess the false positives guarantees of caSPiTa using pseudo-artificial datasets. Starting from a real dataset, we created its h-th order generative null model, which we used to generate random datasets using the POG strategy. Each random dataset is then a bag of paths of a given length $k > h$ that does not contain any significant path of length k (since they have been generated in accordance with the generative null model). We then executed caSPiTa on each random dataset, with parameters h and k corresponding to the ones used to generate the random dataset, and we checked whether caSPiTa reported some paths, which would be false positives by construction. We considered BIKE10 and BIKE20 as starting real datasets, $k \in \{2, \ldots, 5\}$, and $h \in \{1, \ldots, k-1\}$, mining under and over-represented paths. Given a real dataset, we generated 20 random datasets for each combinations of h and k, obtaining a total number of 400 runs for each real dataset. We then estimated the FWER as the fraction of runs with at least one false positive. Table 7.3 (g=1) shows the obtained results. For BIKE10, caSPiTa obtained an estimated FWER of 0.75% with $P = 100$, 1.25% with $P = 1000$, and 0.5% with $P = 10000$. Instead, for BIKE20, it obtained an estimated FWER of 2.25% with $P = 100$, 1.75% with $P = 1000$, and 3.25% with $P = 10000$. These results show that the false positives guarantees of caSPiTa are even better than the theoretical ones, which are $\leq 5\%$ using $\alpha = 0.05$, and that $P = 100$ is enough to obtain such guarantees. Using these random datasets, we also made a comparison with HYPA. Let us remember that HYPA employs a fixed threshold β to flag as anomalous a path, without any theoretical guarantees. We used $\beta \in \{0.00001, 0.001, 0.05\}$, which are, respectively, the minimum, Table 7.3 g-FWER estimates for g-caSPiTa with pseudo-artificial datasets obtained from BIKE10 and BIKE20. The Table reports: P: number of random datasets for the WY method; g: number of false positives considered in the g-FWER; for each real datasets, BIKE10 and BIKE 20, g-FWER (%): estimate of the g-FWER in percentage.

	BIKE10				BIKE20			
	g-FWER (%)				g-FWER (%)			
P\\g	1	2	5	10	1	2	5	10
100	0.75	0.50	0.75	0.25	2.25	1.25	0.25	0.50
1000	1.25	0.50	0.75	0.25	1.75	0.25	1.25	0.75
10000	0.50	1.25	0.10	0.10	3.25	1.25	0.25	0.50

the most commonly used, and the maximum value used in [40], and $k \in \{2, \ldots, 5\}$. (For h, HYPA always considers $h = k - 1$, thus we only used this value.) For BIKE10, it obtained an estimated FWER of 40.63% with $\beta = 0.05$, 15.63% with $\beta = 0.001$, and 0.0% with $\beta = 0.00001$. Instead, for BIKE20, it obtained an estimated FWER of 60.63% with $\beta = 0.05$, 32.50% with $\beta = 0.001$, and 1.25% with $\beta = 0.00001$. These results show that HYPA is able to return anomalous paths achieving low FWER with the correct threshold, but also emphasize the importance of having a strategy, as the one that we employ, to compute such a threshold in an automatic way, since the usage of fixed thresholds may lead to many spurious discoveries, or to a low statistical power.

Finally, considering the same pseudo-artificial datasets, we assessed the false positives guarantees of g-caSPiTa with $g \in \{2, 5, 10\}$. We executed g-caSPiTa in each random dataset and then we estimated the g-FWER as the fraction of runs with at least g false positives. Table 7.3 shows all the obtained results. Similarly to what has been observed with 1-caSPiTa, i.e., the original version of caSPiTa, the results show that the false positives guarantees of g-caSPiTa are even better than the theoretical ones since all the g-FWER estimates are (far) below 5%, and $P = 100$ is enough to obtain them.

7.6 Results with Real Datasets

We then executed (g-)caSPiTa on some real datasets, i.e., BIKE, WIKI, and FLIGHT. Tables 7.4, 7.5, and 7.6 report the results obtained with $P = 100$, $k \in \{2, \ldots, 5\}$, $h \in \{1, \ldots, k-1\}$, and $g \in \{1, 2, 5, 10\}$. For the computational time, we only reported the values for $g = 1$, since the others are analogous. Considering $g = 1$, for all the datasets, and for almost all combinations of parameters, caSPiTa reported some significant paths. It is interesting to notice that the number of over-represented paths is almost always (some

Table 7.4 g-caSPiTa results with BIKE. k: paths length; h: order of the null model; $|\mathcal{W}|$: number of distinct paths of length k; T: number of total paths of length k; for each g, number of reported significant paths: over- (+) and under- (−) represented; Time (s): execution time in seconds for $g = 1$.

k	h	$\|\mathcal{W}\|$	T	$g=1$		$g=2$		$g=5$		$g=10$		Time (s)
				+	−	+	−	+	−	+	−	
2	1	90.5K	252K	197	28	256	47	374	84	453	98	150
3	1	172K	221K	118	40	159	60	226	85	311	117	350
	2			1	8	4	17	7	29	10	38	375
4	1	179K	195K	80	15	127	22	153	40	184	65	662
	2			10	7	14	15	19	25	21	33	639
	3			0	3	0	4	0	12	0	15	713
5	1	166K	174K	71	6	98	6	115	9	130	19	855
	2			17	3	19	5	31	10	39	16	514
	3			0	1	1	3	1	3	2	9	458
	4			0	1	0	2	0	4	1	7	365

Table 7.5 g-caSPiTa results with WIKI. See Table 7.4 for the meaning of the reported values.

k	h	$\|\mathcal{W}\|$	T	$g=1$		$g=2$		$g=5$		$g=10$		Time (s)
				+	−	+	−	+	−	+	−	
2	1	155K	244K	160	53	222	82	346	134	424	175	264
3	1	169K	194K	219	6	334	16	443	27	549	34	384
	2			8	12	11	13	22	25	27	34	495
4	1	139K	147K	193	1	319	2	425	3	535	3	765
	2			16	6	20	8	39	10	51	16	747
	3			2	2	4	6	6	12	7	12	594
5	1	106K	108K	113	0	154	0	243	0	327	0	1.03K
	2			7	0	9	0	18	1	39	1	371
	3			0	1	0	2	0	5	4	8	269
	4			0	0	0	0	1	1	1	1	211

Table 7.6 g-caSPiTa results with FLIGHT. See Table 7.4 for the meaning of the reported values.

k	h	$\|\mathcal{W}\|$	T	$g=1$		$g=2$		$g=5$		$g=10$		Time (s)
				$+$	$-$	$+$	$-$	$+$	$-$	$+$	$-$	
2	1	574K	11.1M	96.5K	16.4K	102K	19.2K	111K	23.2K	115K	26.1K	5.73K
3	1	849K	5.16M	132K	36	140K	50	144K	75	149K	92	10.0K
	2			128K	1.63K	134K	2.13K	140K	2.81K	144K	3.25K	13.3K
4	1	406K	530K	19.2K	0	22.0K	0	25.1K	0	26.6K	0	2.85K
	2			10.4K	30	14.8K	34	19.2K	43	22.0K	52	1.99K
	3			3.11K	19	3.84K	24	5.00K	50	5.95K	66	1.52K
5	1	127K	155K	6.66K	0	7.92K	0	9.39K	0	10.1K	0	43.9K
	2			4.28K	0	5.16K	0	6.35K	0	7.07K	0	1.31K
	3			1.80K	5	2.32K	7	3.30K	8	4.01K	8	723
	4			884	2	1.30K	2	1.87K	3	2.23K	4	638

orders of magnitude) greater than the number of under-represented paths. In addition, for a fixed value of k, the number of over-represented paths always decreases considering higher values of h. The number of under-represented paths, instead, always decreases for BIKE, while increases and then decreases for FLIGHT and WIKI, highlighting different substructures in the three underlying networks. The computational time ranges from under 3 minutes (BIKE with $k = 2$, $h = 1$) to over 12 hours (FLIGHT with $k = 5$, $h = 1$). Let us note that FLIGHT has over $17M$ transactions, but we are still able to analyze it with a reasonable running time. The combination $k = 5$ and $h = 1$ is always the most expensive from a computational point of view, since it requires to generate longer paths and also to analyze a large number of vertices to compute the probabilities of Equation 7.6. Overall, these results show that caSPiTa is able to efficiently mine significant paths from real datasets, with feasible computational time even in huge datasets. Considering $g > 1$, for all the datasets, and for almost all combinations of parameters, g-caSPiTa reported more significant paths, both over and under-represented, w.r.t. the paths reported by the original version of caSPiTa. In particular, for BIKE, g-caSPiTa with $g = 2$ reported on average 77% more paths than caSPiTa,[6] 188% more with $g = 5$, and 330% more with $g = 10$. For WIKI, instead, g-caSPiTa with $g = 2$ reported on average 70% more paths than caSPiTa, 194% more with $g = 5$, and 286% more with $g = 10$. Finally, for FLIGHT, g-caSPiTa with $g = 2$ reported on average 22% more paths than caSPiTa, 61% more with $g = 5$, and 86% more with $g = 10$. Let us note that even if we were allowing very few false positives, the number of reported paths increased considerably.

[6] We only considered combinations of parameters for which caSPiTa reported at least one significant path.

Table 7.7 caSPiTa results on BIKE considering different datasets. The Table reports: k: paths length; h: order of the null model; $|\mathcal{W}|$: number of distinct paths of length k; T: number of total paths of length k; $(+)$: reported over-represented paths; $(-)$: reported under-represented paths.

$k = h$	$\lvert\mathcal{W}\rvert$	T	$+$	$-$
1	5.79K	68.0K	256	120
2	5.51K	12.2K	30	9
3	1.19K	2.45K	14	1
4	436	786	4	0
5	124	203	0	0

Overall, these results show that to control the g-FWER is a valid strategy to increase the statistical power for those applications which can tolerate a small number of false positives.

7.6.1 Analysis of BIKE with caSPiTa

In this Section, we provide a brief analysis of some paths returned by caSPiTa from BIKE. The over-represented path of length 2 with the lowest p-value and highest number of occurrences is a path which starts and ends in “Ocean Front Walk & Navy" located in Venice Beach. The fact that this path is over-represented indicates that people tend to leave and then come back to this place, instead of moving to other parts of the city. For example, such a pattern may capture the fact that people leave the beach to buy some food and then immediately come back. Instead, the under-represented path of length 2 with the lowest p-value is a path which starts from “Union Station West Portal”, goes to “Main & 1st", and then comes back. “Main & 1st" is located near the Los Angeles City Hall, the center of the government of the city, while “Union Station West Portal” is near the Union Station, the main railway station of the city. The fact that this path is under represented is probably due to the fact that a lot of people move from the station to the city hall, and vice-versa, but in particular moments of the day, i.e., in the morning and in the evening. Thus, even if the two direct links are very popular, it is uncommon to see this entire path. These are only two example of paths mined by caSPiTa, but they highlight its capability in detecting real life trends.

Finally, we investigated the capability of caSPiTa in mining significant paths in the scenario in which the generative null model is created considering a different dataset (see Section 7.4). Using the procedure to generate BIKE described above, we generated a new dataset, NEWBIKE, considering the 2020 data from the same website. We then used the original BIKE dataset to create the h-th order generative null model and we tested on it the significance of length k paths mined from NEWBIKE. Given the pandemic situation that involved the world, one may be interested in finding changes in the habits of the people defining a model based on paths traveled in 2019 to test paths traveled in 2020. Table 7.7 reports the results obtained with $k = h \in \{1, \ldots, 5\}$ and $P = 100$. (For NEWBIKE, we only considered full transactions that can be generated by the generative null model obtained from BIKE.) Again, it is possible to notice that caSPiTa returned paths for almost all combinations of parameters, and that the number of over-represented paths is always higher than the number of under-represented paths. Overall, these results demonstrate that caSPiTa is able to mine paths also in such a scenario.

Chapter 8

Conclusions

In this Chapter, we summarize the contributions of this Thesis and discuss some possible future research directions. In this Thesis, we contributed novel scalable and rigorous results on the mining of patterns in sequential data in different scenarios.

In Chapter 3, we studied the task of *mining frequent sequential patterns* through *sampling*. In particular, we studied the Vapnik-Chervonenkis (VC) dimension of sequential patterns, introducing a novel and efficient upper bound on such a measure. We then developed the first sampling-based algorithm to mine rigorous approximations of the frequent sequential patterns, with guarantees defined in terms of false positives or false negatives, that hinges on the novel bound on the VC-dimension. Our extensive experimental evaluation showed that our sampling-based algorithm for mining frequent sequential patterns produces high-quality approximations using samples that are small fractions of the whole datasets, thus vastly speeding up the frequent sequential pattern mining task on very large datasets.

In Chapter 4, we introduced the task of *mining true frequent sequential patterns*, defined as sequential patterns that are frequently generated by the underlying generative process of the data. In particular, we developed an algorithm to mine rigorous approximations of the true frequent sequential patterns, with guarantees defined in terms of false positives or false negatives, using results on the VC-dimension introduced in Chapter 3. Our extensive experimental evaluation showed that techniques like the one we introduced in this Chapter are necessary for mining frequent sequential patterns w.r.t. an underlying generative process avoiding false positives or false negatives, and that our algorithm provides high-quality approximations even better than guaranteed by its theoretical analysis.

In Chapter 5, we introduced the task of *mining statistically robust patterns* from a *sequence of datasets*, defined as patterns whose probabilities w.r.t. the underlying generative processes of the data follow well specified trends through the sequence of datasets, which naturally arises in several applications. We provided a general framework for such a problem and developed gRosSo, an algorithm to identify approximations of the statistically robust patterns with probabilistic guarantees on false discoveries or false negatives. We then applied it to identify statistically robust sequential patterns and statistically robust itemsets, using previous results on the VC-dimension of sequential patterns and itemsets. Our extensive experimental evaluation showed that gRosSo significantly improves over the naïve approach which ignores the uncertainty in the data, and that it identified interesting patterns in real datasets, providing high-quality approximations.

In Chapter 6, we introduced the task of *mining statistically significant sequential patterns*, defined as sequential patterns that appear more frequently than expected under a null model that randomizes the order of the itemsets in the dataset. We proposed three efficient strategies to generate random datasets from the aforementioned null model and developed ProMiSe, an algorithm for mining significant sequential patterns employing the generation strategies, within the Westfall-Young (WY) method, to properly control the Family-Wise Error Rate (FWER). We then developed I-ProMiSe, an alternative, more efficient but less statistically powerful version of ProMiSe, for mining significant sequential patterns from massive datasets employing previous results on the VC-dimension of sequential patterns. Our extensive experimental evaluation showed that ProMiSe and I-ProMiSe efficiently extract significant sequential patterns in real datasets while correctly controlling the FWER.

In Chapter 7, we introduced the task of *mining statistically significant paths* in *time series data* from an *unknown network*, defined as patterns that occur more or less than expected given the distribution of the underlying network that generated the time series. We proposed an appropriate null model of the data, based on the concept of De Bruijn graph, and described two strategies to generate random time series data from it. We then developed caSPiTa, an algorithm for mining statistically significant paths (over- or under-represented) employing the generation strategies, within the WY method, to properly control the FWER. We developed an alternative version of caSPiTa, g-caSPiTa, for mining statistically significant paths while controlling the generalized FWER (g-FWER), in order to increase the statistical power of caSPiTa tolerating a few false positives. Our extensive experimental evaluation showed that (g-)caSPiTa is able to efficiently mine large sets of significant paths from real datasets, while correctly controlling the (generalized) FWER.

There are many possible extensions of the contributions of this work and new directions for future research. For the scenario of mining patterns from samples from unknown probability distributions, i.e., Chapters 4, 5, and 6 (for I-ProMiSe), while we employed the VC-dimension to bound the maximum deviation, any uniform convergence bound can be used in our frameworks. In particular, recent results based on the Monte Carlo empirical Rademacher averages [55, 56] derived sharp, data-dependent, uniformly valid confidence bounds on the expectations of sets of functions from random samples, that can be applied to our scenarios in order to increase the statistical power.

For the statistically robust pattern mining task, while we applied our framework for mining statistically robust sequential patterns and itemsets, other pattern mining tasks can be taken into consideration, such as subgroup discovery [6, 7, 36] and subgraph mining [31]. Other interesting directions are to consider the problem in a streaming setting for the data [39, 1] and to take into account more types of trends, than emerging, descending and stable, for the true frequencies of the patterns.

For the scenario of mining significant patterns, i.e., Chapters 6 and 7, while we focused on bounding the FWER (or g-FWER), different approaches would be to bound the False Discovery Rate (FDR) [8, 9], defined as the expected ratio of false discoveries among all reported patterns, or the False Discovery Proportion (FDP) [71, 72], defined as the probability of rejecting a set of hypotheses with a fraction of false discoveries higher than a given threshold. In such a direction, the work by Komiyama et al. [38] proposed a strategy to correctly control the FDR in the significant pattern mining scenario introducing the notion of "quasi-testability", which allows to reduce the number of hypotheses to test by removing the ones that are very unlikely to be significant. By combining such a notion with the step-up method [30], their approach successfully control the FDR. An interesting question is whether their approach can be extended to work in our scenarios. Finally, another interesting direction is to derive efficient strategies to identify the top-k significant sequential patterns and top-k significant paths, as already done in other significant pattern mining scenarios [60].

References

[1] Aggarwal CC (2007) Data streams: models and algorithms. Vol. 31, Springer Science & Business Media.

[2] Agrawal R, Imieliński T, Swami A (1993) Mining association rules between sets of items in large databases. In: Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, pp. 207–216.

[3] Agrawal R, Srikant R (1995) Mining sequential patterns. In: Proceedings of the eleventh International Conference on Data Engineering, IEEE, pp. 3–14.

[4] Ahmed NK, Neville J, Rossi RA, Duffield N (2015) Efficient graphlet counting for large networks. In: 2015 IEEE International Conference on Data Mining, IEEE, pp. 1–10.

[5] Akoglu L, Tong H, Koutra D (2015) Graph based anomaly detection and description: a survey. Data Mining and Knowledge Discovery, 29(3), 626–688.

[6] Belfodil A, Belfodil A, Bendimerad A, Lamarre P, Robardet C, Kaytoue M, Plantevit M (2019) FSSD - A fast and efficient algorithm for subgroup set discovery. In: 2019 IEEE International Conference on Data Science and Advanced Analytics (DSAA), IEEE, pp. 91–99.

[7] Belfodil A, Belfodil A, Kaytoue M (2018) Anytime subgroup discovery in numerical domains with guarantees. In: Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer, pp. 500–516.

[8] Benjamini Y, Hochberg Y (1995) Controlling the false discovery rate: a practical and powerful approach to multiple testing. Journal of the Royal Statistical Society: series B (Methodological), 57(1), 289–300.

[9] Benjamini Y, Yekutieli D (2001) The control of the false discovery rate in multiple testing under dependency. Annals of Statistics, pp. 1165–1188.

[10] Boley M, Horváth T, Wrobel S (2009) Efficient discovery of interesting patterns based on strong closedness. Statistical Analysis and Data Mining: The ASA Data Science Journal, 2(5-6), 346–360.

[11] Bonferroni C (1936) Teoria statistica delle classi e calcolo delle probabilita. Pubblicazioni del R Istituto Superiore di Scienze Economiche e Commericiali di Firenze, 8, 3–62.

[12] Boucheron S, Bousquet O, Lugosi G (2005) Theory of classification: A survey of some recent advances. ESAIM: Probability and Statistics, 9, 323–375.

[13] Cobb GW, Chen YP (2003) An application of markov chain monte carlo to community ecology. The American Mathematical Monthly, 110(4), 265–288.

[14] Dong G, Li J (1999) Efficient mining of emerging patterns: Discovering trends and differences. In: Proceedings of the fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 43–52.

[15] Egho E, Gay D, Boullé M, Voisine N, Clérot F (2017) A user parameter-free approach for mining robust sequential classification rules. Knowledge and Information Systems, 52(1), 53–81.

[16] Egho E, Raïssi C, Calders T, Jay N, Napoli A (2015) On measuring similarity for sequences of itemsets. Data Mining and Knowledge Discovery, 29(3), 732–764.

[17] Esling P, Agon C (2012) Time-series data mining. ACM Computing Surveys (CSUR), 45(1), 1–34.

[18] Fournier-Viger P, Lin JCW, Gomariz A, Gueniche T, Soltani A, Deng Z, Lam HT (2016) The spmf open-source data mining library version 2. In: Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer, pp. 36–40.

[19] Fournier-Viger P, Lin JCW, Nkambou R, Vo B, Tseng VS (2019) High-utility pattern mining. Springer.

[20] Fournier-Viger P, Lin JCW, Truong-Chi T, Nkambou R (2019) A survey of high utility itemset mining. In: High-utility pattern mining, Springer, pp. 1–45.

[21] Gionis A, Mannila H, Mielikäinen T, Tsaparas P (2007) Assessing data mining results via swap randomization. ACM Transactions on Knowledge Discovery from Data (TKDD), 1(3), 14–es.

[22] Gobbi A, Iorio F, Dawson KJ, Wedge DC, Tamborero D, Alexandrov LB, Lopez-Bigas N, Garnett MJ, Jurman G, Saez-Rodriguez J (2014) Fast randomization of large genomic datasets while preserving alteration counts. Bioinformatics, 30(17), i617–i623.

[23] Gupta M, Gao J, Aggarwal CC, Han J (2013) Outlier detection for temporal data: A survey. IEEE Transactions on Knowledge and Data Engineering, 26(9), 2250–2267.

[24] Gwadera R, Crestani F (2010) Ranking sequential patterns with respect to significance. In: Pacific-Asia Conference on Knowledge Discovery and Data Mining, Springer, pp. 286–299.

[25] Hämäläinen W, Nykänen M (2008) Efficient discovery of statistically significant association rules. In: 2008 Eighth IEEE International Conference on Data Mining, IEEE, pp. 203–212.

[26] Hämäläinen W, Webb GI (2019) A tutorial on statistically sound pattern discovery. Data Mining and Knowledge Discovery, 33(2), 325–377.

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

[28] Han J, Pei J, Mortazavi-Asl B, Chen Q, Dayal U, Hsu MC (2000) Freespan: frequent pattern-projected sequential pattern mining. In: Proceedings of the sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 355–359.

[29] Han J, Pei J, Yin Y, Mao R (2004) Mining frequent patterns without candidate generation: A frequent-pattern tree approach. Data Mining and Knowledge Discovery, 8(1), 53–87.

[30] Hochberg Y (1988) A sharper bonferroni procedure for multiple tests of significance. Biometrika, 75(4), 800–802.

[31] Jiang C, Coenen F, Zito M (2013) A survey of frequent subgraph mining algorithms. The Knowledge Engineering Review, 28(1), 75–105.

[32] Keogh E, Kasetty S (2003) On the need for time series data mining benchmarks: a survey and empirical demonstration. Data Mining and Knowledge Discovery, 7(4), 349–371.

[33] Keogh E, Lin J (2005) Clustering of time-series subsequences is meaningless: implications for previous and future research. Knowledge and Information Systems, 8(2), 154–177.

[34] Keogh E, Lonardi S, Chiu BC (2022) Finding surprising patterns in a time series database in linear time and space. In: Proceedings of the eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 550–556.

[35] Kirsch A, Mitzenmacher M, Pietracaprina A, Pucci G, Upfal E, Vandin F (2012) An efficient rigorous approach for identifying statistically significant frequent itemsets. Journal of the ACM (JACM), 59(3), 1–22.

[36] Klösgen W (1992) Problems for knowledge discovery in databases and their treatment in the statistics interpreter explora. International Journal of Intelligent Systems, 7(7), 649–673.

[37] Knuth DE (2014) Art of computer programming, volume 2: Seminumerical algorithms. Addison-Wesley Professional.

[38] Komiyama J, Ishihata M, Arimura H, Nishibayashi T, Minato SI (2017) Statistical emerging pattern mining with multiple testing correction. In: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 897–906.

[39] Krempl G, Žliobaite I, Brzeziński D, Hüllermeier E, Last M, Lemaire V, Noack T, Shaker A, Sievi S, Spiliopoulou M, et al. (2014) Open challenges for data stream mining research. ACM SIGKDD Explorations Newsletter, 16(1), 1–10.

[40] LaRock T, Nanumyan V, Scholtes I, Casiraghi G, Eliassi-Rad T, Schweitzer F (2020) Hypa: Efficient detection of path anomalies in time series data on networks. In: Proceedings of the 2020 SIAM International Conference on Data Mining, SIAM, pp. 460–468.

[41] Lehmann EL, Romano JP (2012) Generalizations of the familywise error rate. In: Selected Works of EL Lehmann, Springer, pp. 719–735.

[42] Lemmerich F, Becker M, Singer P, Helic D, Hotho A, Strohmaier M (2016) Mining subgroups with exceptional transition behavior. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 965–974.

[43] Li Y, Long PM, Srinivasan A (2001) Improved bounds on the sample complexity of learning. Journal of Computer and System Sciences, 62(3), 516–527.

[44] Lin J, Keogh E, Lonardi S, Lankford JP, Nystrom DM (2004) Visually mining and monitoring massive time series. In: Proceedings of the tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 460–469.

[45] Llinares-López F, Sugiyama M, Papaxanthos L, Borgwardt K (2015) Fast and memory-efficient significant pattern mining via permutation testing. In: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 725–734.

[46] Löffler M, Phillips JM (2009) Shape fitting on point sets with probability distributions. In: European Symposium on Algorithms, Springer, pp. 313–324.

[47] Low-Kam C, Raïssi C, Kaytoue M, Pei J (2013) Mining statistically significant sequential patterns. In: 2013 IEEE 13th International Conference on Data Mining, IEEE, pp. 488–497.

[48] Mampaey M, Vreeken J, Tatti N (2012) Summarizing data succinctly with the most informative itemsets. ACM Transactions on Knowledge Discovery from Data (TKDD), 6(4), 1–42.

[49] Mendes LF, Ding B, Han J (2008) Stream sequential pattern mining with precise error bounds. In: 2008 Eighth IEEE International Conference on Data Mining, IEEE, pp. 941–946.

[50] Milo R, Kashtan N, Itzkovitz S, Newman ME, Alon U (2003) On the uniform generation of random graphs with prescribed degree sequences. arXiv preprintcond-mat/0312028.

[51] Minato SI, Uno T, Tsuda K, Terada A, Sese J (2014) A fast method of statistical assessment for combinatorial hypotheses based on frequent itemset enumeration. In: Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer, pp. 422–436.

[52] Mitzenmacher M, Upfal E (2017) Probability and computing: Randomization and probabilistic techniques in algorithms and data analysis. Cambridge University Press.

[53] Noble CC, Cook DJ (2003) Graph-based anomaly detection. In: Proceedings of the ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp.631–636.

[54] Pei J, Han J, Mortazavi-Asl B, Wang J, Pinto H, Chen Q, Dayal U, Hsu MC (2004) Mining sequential patterns by pattern-growth: The prefixspan approach. IEEE Transactions on Knowledge and Data Engineering, 16(11), 1424–1440.

[55] Pellegrina L (2020) Sharper convergence bounds of monte carlo rademacher averages through self-bounding functions. arXiv preprint arXiv:2010.12103 .

[56] Pellegrina L, Cousins C, Vandin F, Riondato M (2020) Mcrapper: Monte-carlo rademacher averages for poset families and approximate pattern mining. In: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 2165–2174.

[57] Pellegrina L, Pizzi C, Vandin F (2020) Fast approximation of frequent k-mers and applications to metagenomics. Journal of Computational Biology, 27(4), 534–549.

[58] Pellegrina L, Riondato M, Vandin F (2019) Hypothesis testing and statistically-sound pattern mining. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 3215–3216.

[59] Pellegrina L, Riondato M, Vandin F (2019) Spumante: Significant pattern mining with unconditional testing. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 1528–1538.

[60] Pellegrina L, Vandin F (2020) Efficient mining of the most significant patterns with permutation testing. Data Mining and Knowledge Discovery, 34, 1201–1234.

[61] Pietracaprina A, Riondato M, Upfal E, Vandin F (2010) Mining top-k frequent itemsets through progressive sampling. Data Mining and Knowledge Discovery, 21(2), 310–326.

[62] Pinxteren S, Calders T (2021) Efficient permutation testing for significant sequential patterns. In: Proceedings of the 2021 SIAM International Conference on Data Mining (SDM), SIAM, pp. 19–27.

[63] Preti G, De Francisci Morales G, Riondato M (2021) Maniacs: Approximate mining of frequent subgraph patterns through sampling. In: Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining, pp. 1348–1358.

[64] Raïssi C, Calders T, Poncelet P (2008) Mining conjunctive sequential patterns. Data Mining and Knowledge Discovery, 17(1), 77–93.

[65] Raïssi C, Poncelet P (2007) Sampling for sequential pattern mining: From static databases to data streams. In: Seventh IEEE International Conference on Data Mining (ICDM 2007), IEEE, pp. 631–636.

[66] Riondato M, Upfal E (2014) Efficient discovery of association rules and frequent itemsets through sampling with tight performance guarantees. ACM Transactions on Knowledge Discovery from Data (TKDD), 8(4), 1–32.

[67] Riondato M, Upfal E (2015) Mining frequent itemsets through progressive sampling with rademacher averages. In: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1005–1014.

[68] Riondato M, Vandin F (2014) Finding the true frequent itemsets. In: Proceedings of the 2014 SIAM International Conference on Data Mining, SIAM, pp. 497–505.

[69] Riondato M, Vandin F (2020) Misosoup: Mining interesting subgroups with sampling and pseudodimension. ACM Transactions on Knowledge Discovery from Data (TKDD), 14(5), 1–31.

[70] Roddick JF, Hornsby K, Spiliopoulou M (2000) An updated bibliography of temporal, spatial, and spatio-temporal data mining research. In: International Workshop on Temporal, Spatial, and Spatio-Temporal Data Mining, Springer, pp. 147–163.

[71] Romano JP, Shaikh AM (2006) On stepdown control of the false discovery proportion. In: Optimality, Institute of Mathematical Statistics, pp. 33–50.

[72] Romano JP, Shaikh AM (2006) Stepup procedures for control of generalizations of the familywise error rate. The Annals of Statistics, 34(4), 1850–1873.

[73] Santoro D, Tonon A, Vandin F (2020) Mining sequential patterns with VC-dimension and rademacher complexity. Algorithms, 13(5), 123.

[74] Scholtes I (2017) When is a network a network? Multi-order graphical model selection in pathways and temporal networks. In: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1037–1046.

[75] Servan-Schreiber S, Riondato M, Zgraggen E (2020) Prosecco: Progressive sequence mining with convergence guarantees. Knowledge and Information Systems, 62(4), 1313–1340.

[76] Shalev-Shwartz S, Ben-David S (2014) Understanding machine learning: From theory to algorithms. Cambridge University Press.

[77] Srikant R, Agrawal R (1996) Mining sequential patterns: Generalizations and performance improvements. In: International Conference on Extending Database Technology, Springer, pp. 1–17.

[78] Tatti N, Mampaey M (2010) Using background knowledge to rank itemsets. Data Mining and Knowledge Discovery, 21(2), 293–309.

[79] Terada A, Kim H, Sese J (2015) High-speed westfall-young permutation procedure for genome-wide association studies. In: Proceedings of the 6th ACM Conference on Bioinformatics, Computational Biology and Health Informatics, pp. 17–26.

[80] Terada A, Okada-Hatakeyama M, Tsuda K, Sese J (2013) Statistical significance of combinatorial regulations. Proceedings of the National Academy of Sciences, 110(32), 12996–13001.

[81] Terada A, Tsuda K, Sese J (2013) Fast westfall-young permutation procedure for combinatorial regulation discovery. In: 2013 IEEE International Conference on Bioinformatics and Biomedicine, IEEE, pp. 153–158.

[82] Toivonen H (1996) Sampling large databases for association rules. In: VLDB, Vol. 96, pp. 134–145.

[83] Tonon A, Vandin F (2019) Permutation strategies for mining significant sequential patterns. In: 2019 IEEE International Conference on Data Mining (ICDM), IEEE, pp. 1330–1335.

[84] Tonon A, Vandin F (2020) Grosso: Mining statistically robust patterns from a sequence of datasets. In: 2020 IEEE International Conference on Data Mining (ICDM), IEEE, pp. 551–560.

[85] Tonon A, Vandin F (2021) Caspita: Mining statistically significant paths in time series data from an unknown network. In: 2021 IEEE International Conference on Data Mining (ICDM), IEEE, pp. 639-648.

[86] Vapnik VN (1999) The nature of statistical learning theory, Springer Science & Business Media.

[87] Vapnik VN, Chervonenkis AY (2015) On the uniform convergence of relative frequencies of events to their probabilities. In: Measures of Complexity, Springer, pp. 11–30.

[88] Wang J, Han J, Li C (2007) Frequent closed sequence mining without candidate maintenance. IEEE Transactions on Knowledge and Data Engineering, 19(8), 1042–1056.

[89] Webb GI (2006) Discovering significant rules. In: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 434–443.

[90] Webb GI (2007) Discovering significant patterns. Machine Learning, 68(1), 1–33.

[91] Webb GI (2008) Layered critical values: a powerful direct-adjustment approach to discovering significant patterns. Machine Learning, 71(2-3), 307–323.

[92] Wei L, Keogh E (2006) Semi-supervised time series classification. In: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 748–753.

[93] Weiss GM (2004) Mining with rarity: a unifying framework. ACM SIGKDD Explorations Newsletter, 6(1), 7–19.

[94] West R, Leskovec J (2012) Human wayfinding in information networks. In: Proceedings of the 21st International Conference on World Wide Web, pp. 619–628.

[95] Westfall PH, Young SS (1993) Resampling-based multiple testing: Examples and methods for p-value adjustment. Vol. 279, John Wiley & Sons.

[96] Yan X, Han J, Afshar R (2003) Clospan: Mining closed sequential patterns in large datasets. In: Proceedings of the 2003 SIAM International Conference on Data Mining, SIAM, pp. 166–177.

[97] Zhu F, Yan X, Han J, Philip SY, Cheng H (2007) Mining colossal frequent patterns by core pattern fusion. In: 2007 IEEE 23rd International Conference on Data Engineering, IEEE, pp. 706–715.