Noname manuscript No.
(will be inserted by the editor)

Efficient Mining of the Most Significant Patterns with Permutation Testing

Leonardo Pellegrina · **Fabio Vandin**

Received: date / Accepted: date

Abstract The extraction of patterns displaying significant association with a class label is a key data mining task with wide application in many domains. We introduce and study a variant of the problem that requires to mine the top-k statistically significant patterns, thus providing tight control on the number of patterns reported in output. We develop TopKWY, the first algorithm to mine the top-k significant patterns while rigorously controlling the family-wise error rate of the output, and provide theoretical evidence of its effectiveness. TopKWY crucially relies on a novel strategy to explore statistically significant patterns and on several key implementation choices, which may be of independent interest. Our extensive experimental evaluation shows that TopKWY enables the extraction of the most significant patterns from large datasets which could not be analyzed by the state-of-the-art. In addition, TopKWY improves over the state-of-the-art even for the extraction of *all* significant patterns.

Keywords Statistical pattern mining; hypothesis testing; top-k patterns.

A preliminary version of this work appeared in the proceedings of ACM KDD'18 as (Pellegrina and Vandin, 2018).

Leonardo Pellegrina
Department of Information Engineering
Università di Padova
Via G. Gradenigo 6/B, Padova, 35131 (IT)
E-mail: pellegri@dei.unipd.it

Fabio Vandin
Department of Information Engineering
Università di Padova
Via G. Gradenigo 6/B, Padova, 35131 (IT)
E-mail: fabio.vandin@unipd.it
Corresponding author

1 Introduction

Frequent patterns mining is one of the fundamental primitives in data mining, with applications in a large number of domains, ranging from market basket analysis to biology and medicine (Han et al., 2007). In its original definition (Agrawal et al., 1993) it requires to identify patterns that appear in a fraction at least σ of all the transactions of a transactional dataset. *Significant pattern mining* (Dong and Bailey, 2012; Hämäläinen and Webb, 2019; Pellegrina et al., 2019a) is an extension of the problem in which each transaction is assigned a binary class label and the goal is to identify patterns having *significant association* with one of the class labels. Significance is commonly assessed using a statistical test (e.g., Fisher exact test (Fisher, 1922)), that provides a p-value quantifying the probability that the association observed in real data arises due to chance alone.

Significant pattern mining is crucial in many applications, providing additional information w.r.t. mining patterns that are frequent in the entire dataset: in market basket analysis it serves to identify itemsets that are purchased more frequently by one group of customers than by another one (e.g., married people vs. singles); in social networks, it finds features characterizing users interested in one specific topic; in biology, it identifies sets of genetic variants appearing more frequently in cancer vs normal tissues or in one cancer type vs another one. In all applications, identifying highly reliable associations is of the utmost importance.

One of the critical issues in significant pattern mining is the *multiple hypothesis problem*, due to the huge number of patterns appearing in large datasets. When testing only one pattern, if its p-value is below a fixed threshold α, one can flag the pattern as significant with the guarantee that the probability of *false discovery* (e.g., flagging the pattern as significant when it is not) is bounded by α. However, for a fixed significance threshold α, when d patterns are tested we expect αd of them to have p-value below α even when they are not associated with the class labels. A standard method to correct for multiple hypothesis testing, called *Bonferroni method* (Bonferroni, 1936), is to adjust the significance threshold by dividing α by the number d of tested patterns. This guarantees that the probability of making one or more false discoveries, called *family-wise error rate* (FWER), is bounded by α. However, since the number m of patterns can be huge, this approach results in limited *statistical power*, with very few patterns having p-value passing such small threshold (Webb, 2006, 2007, 2008). A naïve solution for the problem is to limit the number of elements in the patterns to be tested, hindering the ability to identify large significant patterns.

A breakthrough in significant pattern mining is the work by Terada et al. (2013a), that proposes LAMP, the first method to identify significant patterns without limiting their size. LAMP is based on the work by Tarone (1990), which shows that patterns that cannot reach statistical significance, called *untestable*, do not need to be taken into account while correcting for multiple hypothesis testing. Subsequent work by Minato et al. (2014) has improved the

search strategy employed by LAMP to identify testable patterns. Even so, such methods suffer from limited power, due to the use of Bonferroni correction over the large number of testable patterns.

Recently, methods based on the more powerful Westfall-Young permutation procedure (Westfall and Young, 1993) have been proposed, first by Terada et al. (2013b) with FastWY and then by Llinares-López et al. (2015) with Westfall-Young light (WYlight for short). These methods mine several permuted datasets to identify a threshold such that all patterns with p-value below the threshold can be flagged as statistically significant while controlling the FWER. Such methods achieve a higher power than methods based on Bonferroni correction, including LAMP, and the state-of-the-art, WYlight, has proved to be more efficient than FastWY also in terms of runtime and memory.

However, the extraction of significant patterns from large datasets is still challenging, with three crucial issues that are not addressed by currently available methods. First, in several cases the dependency among patterns leads to a huge number of statistically significant patterns even after multiple hypothesis correction. A common approach (used, e.g., by Terada et al. (2013b) and Llinares-López et al. (2015)) to partially alleviate this problem is to consider only closed patterns (Han et al., 2007), discarding patterns with redundant information content in terms of appearance in the dataset and of association with the class label. Even with this restriction the number of significant patterns can be extremely large and when this happens one would like to focus on the most significant ones, without resorting to filtering strategies after the expensive extraction of all significant patterns has been performed. A second and related issue is that current methods work by first identifying the exact corrected threshold for statistical significance, and only subsequently mining the real dataset: when the number of significant patterns is huge, one would like to focus on the most significant ones without the burden of computing the exact significance threshold. Third, all methods may need to process several untestable patterns to identify the correct threshold for significance, resulting in a extremely large running time in particular for datasets with many low frequency patterns. These issues make current methods impractical in many cases, as shown by our experimental evaluation.

1.1 Our Contribution

In this work we focus on the problem of mining the most statistically significant patterns while rigorously controlling the FWER of the returned set of patterns. In particular, in analogy with frequent pattern mining approaches, we focus on extracting the top-k statistically significant patterns. This problem is more challenging than the extraction of top-k frequent patterns, given that statistical significance does not enjoy the anti-monotonicity property w.r.t. to pattern frequency. In this regards, our contributions are:

- we formally define the problem of mining *Top-k Statistically Significant Patterns*. Our definition allows to properly control the size of the output set while providing guarantees on the FWER of the output.
- we design a novel algorithm, called TopKWY, for the problem above, which provides guarantees on the FWER by using the Westfall-Young permutation test procedure. TopKWY adapts to the distribution of significant patterns: it reports *all* significant patterns when their number is small, while it outputs only the most significant patterns when the number of significant patterns is huge.TopKWY is based on a exploration strategy similar to the one used by TopKMiner (Pietracaprina and Vandin, 2007), an efficient algorithm to identify the top-k frequent patterns. We prove that the use of such strategy guarantees that, in contrast to previous approaches, TopKWY will never explore *untestable* patterns.
- we introduce several bounds to prune *untestable* patterns that improve over the bound introduced by LAMP and used in WYlight as well. We show that such bounds can be effectively used within the exploration strategy employed by TopKWY and that it provides a significant speed-up for real datasets.
- we present variants of TopKWY to mine the top-k significant patterns with control of the Generalized Family-Wise Error Rate (g-FWER) or the False Discovery Proportion (FDP), which trade an increase in the size of the output with a (potentially) higher, but still controlled, number of false discoveries. These variants lead to an increase in the *statistical power* in situations where the number of results reported controlling the FWER is low.
- we conduct an extensive experimental evaluation of the use of TopKWY to extract significant itemsets and subgraphs, showing that TopKWY allows the extraction of statistically significant patterns for large datasets while having reasonable memory requirements. Surprisingly, for many datasets TopKWY improves over the state-of-the-art even when it is used to find *all* statistically significant patterns.

1.2 Additional Related Work

Since the introduction of the frequent pattern mining problem (Agrawal et al., 1993), a number of methods have been developed to efficiently extract all frequent patterns (see (Han et al., 2007) for several references). Given that the number of such patterns can be extremely large and that identifying an appropriate frequency threshold to limit the number of frequent patterns is challenging, methods to identify restricted classes of patterns, e.g., closed patterns (Pasquier et al., 1999) or maximal patterns (Bayardo Jr, 1998), have been designed. Methods that directly limit the number of patterns by reporting the k most frequent closed patterns have been designed (Han et al., 2002; Pietracaprina and Vandin, 2007) as well.

Many methods have been developed for *subgroups discovery* (see, e.g., the reviews by Herrera et al. (2011) and by Atzmueller (2015)), that is the task of mining patterns associated with class labels using a *quality score* to quantify the association between a pattern and class labels. Most of the available methods focus on such scores, without assessing the statistical significance of the association or correcting for multiple hypothesis testing. Contrarily to this general trend, a recent work (Li et al., 2014) uses odds ratio as a measure to quantify statistical significance and to reduce redundancies of subgroups, but still does not correct for multiple hypothesis testing. In (Duivesteijn and Knobbe, 2011) multiple hypothesis testing is accounted for using a swap randomization technique (Gionis et al., 2007). The methods we describe in this work can be applied directly to the task of discovering the top-k statistically significant subgroups with proper control of the FWER when the *language* defining the subgroups is composed by conjunctive formulas, and could possibly be adapted and extended to more general languages.

In addition to the contributions for significant pattern mining mentioned above (Terada et al., 2013a; Minato et al., 2014; Terada et al., 2013b, 2015; Llinares-López et al., 2015) (described more in depth in Section 2), recent work has extended the extraction of statistically sound patterns (Hämäläinen and Webb, 2019) in directions that are orthogonal to our contributions, for example searching for statistical dependency rules between itemsets and items (Hämäläinen, 2012), or using layered critical values (Webb, 2008) for correcting for multiple hypothesis testing. Komiyama et al. (2017) have developed efficient techniques for multiple testing correction in the mining of statistical emerging patterns. Terada et al. (2016) and Papaxanthos et al. (2016) have introduced methods to find significant patterns in the presence of covariates.

2 Background and Problem Definition

2.1 Significant Pattern Mining

Let the dataset $\mathcal{D} = \{t_1,t_2,t_3,...,t_n\}$ be a set of n transactions, where each transaction is an element from a domain $\mathcal{F}$. Each transaction t_i is associated to a binary label $c_i \in \{0,1\}$. We denote by n_1 the number of transactions with label 1, and, without loss of generality, we assume that n_1 is the *minority class*, i.e. $n_1 \leqslant n/2$. We define a *pattern* S as an element of $\mathcal{F}$, potentially with some constraints, and for each transaction t we define the binary variable $G(S,t)$ such that $G(S,t) = 1$ if S is *contained* in transaction t and $G(S,t) = 0$ otherwise. For example: in the case of *itemsets*, $\mathcal{F}$ is the set of (non-empty) subsets of a universe of binary features $\mathcal{I}$, and S is an element of $\mathcal{F}$; for *subgraphs*, $\mathcal{F}$ is the set of vertex-labelled graphs, while S is an element of $\mathcal{F}$ with the constraint that S is connected. Given a pattern S, we define its support x_S as the number of transactions containing S, that is $x_S = \sum_{i=1}^{n} G(S,t_i)$. We denote by a_S the number of class 1 transactions containing S. In this work we assume that patterns enjoy the *anti-monotonicity property*, such that for any

Variables	$G(S,t)=1$	$G(S,t)=0$	Row totals
$c=1$	a_S	$n_1 - a_S$	n_1
$c=0$	$x_S - a_S$	$n - n_1 + a_S - x_S$	$n - n_1$
Col totals	x_S	$n - x_S$	n

Fig. 1 2×2 contingency table.

super pattern S' of S (i.e., $S \subset S'$) the support $x_{S'}$ of S' is $x_{S'} \leqslant x_S$. This holds for itemsets, subgraphs, and many other kind of patterns.

The objective of *significant pattern mining* is to find patterns with *significant statistical association* to one of the two classes. In order to quantify the statistical association, a rigorous *statistical test* is performed. Such test assesses the association between the *observations* $G(S,t_1),\ldots,G(S,t_n)$ of variable $G(S,t)$ (describing the presence of S in the transactions of dataset $\mathcal{D}$) and the observed class labels c_i of the transactions, representing observations of the variable $c \in \{0,1\}$ defining the label of a transaction. Since the variables are binary, for a given pattern S the 2×2 contingency table represented in Figure 1 is considered and the popular Fisher exact test (Fisher, 1922) is often employed. Such test considers the *marginals* (x_S, n_1, n) of the contingency table for pattern S to be fixed; under the *null hypothesis* of independence between variables $G(S,t)$ and c, the number a_S of class 1 transactions containing S follows a hypergeometric distribution:

$$\Pr(a_S = a | x_S, n_1, n) = \Pr(a | x_S, n_1, n) = \binom{n_1}{a}\binom{n-n_1}{x_S-a}\Big/\binom{n}{x_S} . \quad (1)$$

Using such distribution, the probability, called *p-value*, of observing an association that is equally or more extreme than the one observed in the data under the null hypothesis can be computed. Smaller p-values indicate more probable associations. The p-value p_S for the pattern S is computed by summing all the probabilities to obtain, under the null hypothesis, contingency tables which are at least as extreme as the one observed in $\mathcal{D}$:

$$p_S = p_S(a_S) = \sum_{k:\Pr(k|x_S,n_1,n) \leqslant Pr(a_S|x_S,n_1,n)} \Pr(k | x_S, n_1, n) \ . \quad (2)$$

2.2 Multiple Hypothesis Testing

When only one pattern S is tested, it can be flagged as *significant* when its p-value is smaller than a *significance threshold* α fixed *a priori*. This guarantees that the probability of a false discovery (i.e., reporting S as significant when it is not) is bounded by α. However, if such approach is used when testing d hypotheses, the expected number of false positives is αd; when d is high, which is typically the case of significant pattern mining, this results in a large number of false positives. Therefore, an appropriate *multiple hypothesis testing correction* of the significance threshold needs to be performed in order to obtain

rigorous guarantees in terms of the number of false associations reported in output.

One common approach is to perform a correction in order to bound the *Family-Wise Error Rate* (FWER), which is defined as the probability of reporting at least one false positive. Let FP be the number of false positives, then: FWER= $\Pr(FP > 0)$. For a given value δ, define $FWER(\delta)$ as the FWER obtained using δ as *corrected* significance threshold, that is by rejecting (i.e., flagging as significant) all null hypotheses (i.e., patterns) with p-value $\leqslant \delta$. Commonly, it is not possible to evaluate $FWER(\delta)$ in closed form. One approach to set δ is to use the Bonferroni correction, setting δ to α/d. Using the union bound, one can easily show that the resulting $FWER(\delta) \leqslant d\delta = \alpha$. The problem with this approach is that when d is high, δ is very close to 0, resulting in low *statistical power* with many false negatives.

To increase the statistical power, more sophisticated techniques have been devised. In particular, one can look for an *optimal* corrected significance threshold δ*, which maximizes the statistical power while keeping the FWER bounded by α:

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

In this regard, a key result from Tarone (1990) is the introduction of *minimum attainable p-value*: if we use a corrected significance threshold δ, patterns whose p-value cannot be $\leqslant \delta$, called *untestable*, do not need to be considered in the multiple-hypothesis correction. Therefore, if we define $k(\delta)$ as the number of patterns having minimum attainable p-value $\leqslant \delta$, then the adjusted Bonferroni correction when threshold δ is used can be written as $\delta = \alpha/k(\delta)$.

LAMP (Terada et al., 2013a) introduced such concepts into the mining of significant patterns. In particular, it exploits the following observation. For a given pattern S, its p-value p_S can be expressed as a function of a_S only. Since the set of allowable values of a_S is finite, i.e. $a_S \in [a_{S,min}, a_{S,max}]$, there exists a minimum attainable p-value $\psi(x_S)$, which depends on x_S, n_1, and n, and which corresponds to the most biased case for equation 2:

$$\psi(x_S) = \min\{p_S(u) \mid a_{S,min} \leqslant u \leqslant a_{S,max}\} \ .$$

Given a pattern S, its support x_S, and a corrected significance threshold δ, if $\psi(x_S) > \delta$ the pattern S cannot be significant; S is therefore called *untestable*. To use such observation for significant pattern mining, Terada et al. (2013a) introduced in LAMP a monotonically decreasing lower bound $\hat{\psi}(x_S)$ on $\psi(x_S)$:

$$\hat{\psi}(x_S) = \begin{cases} \psi(x_S) & 0 \leqslant x_S \leqslant n_1 \\ 1/\binom{n}{n_1} & n_1 < x_S \leqslant n \ . \end{cases}$$

Terada et al. (2013a) showed that identifying a suitable significance threshold δ* translates into finding the maximum support threshold $\sigma_{\max}$ satisfying:

$$\hat{\psi}(\sigma_{\max} - 1) > \frac{\alpha}{k(\sigma_{\max})}$$

and

$$\hat{\psi}(\sigma_{\max}) \leqslant \frac{\alpha}{k(\sigma_{\max}+1)} \ .$$

2.3 Westfall-Young Permutation Testing

LAMP significantly increases the statistical power of the over-conservative standard Bonferroni correction. However, it implicitly assumes that hypotheses are independent, that can result in a loss of power when there is dependence between the hypotheses, as in pattern mining. The Westfall-Young (WY) permutation testing method (Westfall and Young, 1993) is a multiple hypothesis testing procedure capable of addressing this issue. This method performs *random permutations* of the class labels, creating new datasets for which no pattern S is truly associated with the permuted class labels. Since every pattern flagged as significant in the permuted datasets is a false positive, the *joint distribution* of the null hypotheses can be directly estimated, resulting in improved statistical power with respect to LAMP.

In detail, the WY method starts by creating j_p permuted datasets, with j_p sufficiently large (typically in the order of 10^3 or 10^4). Then for every permuted dataset j it computes the minimum p-value $p_{\min}^{(j)}$ over all patterns (hypotheses). Then one can estimate $FWER(\delta)$ as:

$$FWER(\delta) = \frac{1}{j_p} \sum_{j=1}^{j_p} \mathbb{1}[p_{\min}^{(j)} \leqslant \delta]$$

where $\mathbb{1}(\cdot)$ is the indicator function (equal to 1 if its argument is true and 0 otherwise). Given a user provided threshold α for the FWER, the best corrected significance threshold δ^* can then be obtained as $\delta^* = \max_\delta\{FWER(\delta) \leqslant \alpha\}$ with $FWER(\delta)$ estimated as above.

The WY method does not provide an efficient way of computing the set $\{p_{\min}^{(j)}\}_{j=1}^{j_p}$ of minimum p-values. Therefore, a naïve implementation requires to exhaustively test *all* the hypothesis on all the j_p permuted datasets. For significant pattern mining, this means exploring all the patterns appearing in a dataset; since this operation can require exponential time, it is even more challenging to repeat the entire process j_p times, once for every permuted dataset. Terada et al. (2013b) proposed the first efficient implementation, FastWY, of the WY procedure for significant pattern mining. The identification of δ^* is based on a decremental search scheme, which starts with support $\sigma = n$ and iteratively decrements σ until an appropriate condition, guaranteeing that all values $\{p_{\min}^{(j)}\}_{j=1}^{j_p}$ have been computed, is achieved. A more recent method by Terada et al. (2015), HWY, exploits a more efficient mining strategy and parallel computing to accelerate FastWY.

Llinares-López et al. (2015) proposed WYlight to efficiently compute the optimal value δ^* of the corrected significance threshold. The main improvement of WYlight is to avoid the exact computation of all the elements of the

set $\{p_{\min}^{(j)}\}_{j=1}^{j_p}$ and to only produce its exact lower α-quantile. This result is obtained by maintaining an estimate of the α-quantile that is only lowered through the mining process. WYlight performs a depth first exploration of the patterns' search tree (Han et al., 2000) in which each pattern has support less or equal than its parent, and performs only one pattern mining instance, testing one pattern at a time and computing its p-value on all the j_p permuted datasets at the same time. WYlight maintains a threshold σ, initialized at 1, that is raised during the execution of the algorithm pruning patterns whose p-values cannot be in the lower α-quantile of $\{p_{\min}^{(j)}\}_{j=1}^{j_p}$. This is achieved by using the lower bound $\hat{\psi}(\sigma)$ on the minimum obtainable p-value for patterns of support $\leqslant \sigma$, which allows to effectively prune the search tree. However, during the computation of δ^*, some patterns with support $< \psi^{-1}(\delta^*)$ may be processed, due to the depth first procedure considered by WYlight. After computing δ^*, an additional mining of $\mathcal{D}$ is performed (with minimum support threshold $\psi^{-1}(\delta^*)$) to extract the significant patterns with p-value $\leqslant \delta^*$. As shown in (Llinares-López et al., 2015), WYlight significantly improves over FastWY in particular in terms of memory requirements, allowing the extraction of significant patterns from datasets larger than the ones that can be analyzed by FastWY.

2.4 Problem Definition

For a dataset $\mathcal{D}$, let $\delta(\alpha) = \max_\delta\{FWER(\delta) \leqslant \alpha\}$ the threshold obtained through the WY permutation procedure when the bound on the FWER is set to α. Let $p^{(k)}$ be the p-value of the k-th pattern with patterns sorted by (increasing) p-value. Given a dataset $\mathcal{D}$ and user-provided values k and α, our goal is to extract the set $TSP(\mathcal{D}, k, \alpha)$ of top-k statistically significant patterns with FWER $\leqslant \alpha$, defined as:

$$TSP(\mathcal{D}, k, \alpha) = \left\{ S : p_S \leqslant \min\{\delta(\alpha), p^{(k)}\} \right\} \quad .$$

Note that when less than k patterns have p-value below $\delta(\alpha)$, $TSP(\mathcal{D}, k, \alpha)$ contains all such patterns. In addition, according to our definition more than k patterns may be in $TSP(\mathcal{D}, k, \alpha)$, in case many have the same p-value $p^{(k)}$. In particular, for any two patterns S, S' with $S' \subset S$ and $x_{S'} = x_S, a_{S'} = a_S$ we have that $p_S = p_{S'}$. For this reason we restrict our interest only to *closed* patterns, i.e. patterns whose supersets have support *strictly lower* than the pattern itself. Since the definition of closed pattern does not depend on the class labels, restricting to closed patterns does not bias any analysis.

The following result establishes the required guarantees on false positives in $TSP(\mathcal{D}, k, \alpha)$ and it is a direct consequence of the fact that $TSP(\mathcal{D}, k, \alpha)$ is a subset of all the patterns that would be reported using the WY method.

Lemma 1 *The set $TSP(\mathcal{D}, k, \alpha)$ has FWER $\leqslant \alpha$.*

3 TopKWY Algorithm

In this section we present our algorithm TopKWY for mining the set $TSP(\mathcal{D}, k, \alpha)$. We first present its main strategy (Section 3.1) that can be applied to any pattern mining problem. We then analyze TopKWY showing theoretical evidence of the efficiency of its strategy (Section 3.2) and introduce improved bounds on the minimum attainable p-value used by TopKWY (Section 4). We also present extensions of TopKWY (Section 5) to control the *generalized* FWER, to control the False Discovery Proportion (FDP), and to employ different exploration strategies on the tree of candidate patterns. Finally, we introduce some crucial implementation details (Section 6), focusing on the problem of mining significant itemsets and subgraphs.

3.1 Main Strategy

TopKWY combines two key ideas. First, it maintains an estimate of $\delta_m = \min\{\delta(\alpha), p^{(k)}\}$ that is updated during the exploration of the patterns and maintains a corresponding minimum support threshold $\sigma = \psi^{-1}(\delta_m)$ that is raised during the exploration of the patterns. Analogously to the strategy employed by WYlight (Llinares-López et al., 2015) the updates of δ_m and σ depend on α, but in addition TopKWY updates them also depending on the p-values of members of $TSP(\mathcal{D}, k, \alpha)$. Second, the search tree of all possible patterns is explored in order of decreasing support, analogously to the strategy used by TopKMiner (Pietracaprina and Vandin, 2007) for mining top-k frequent patterns, which guarantees that only patterns of support greater or equal to *the final value of* σ (i.e., $\psi^{-1}(\delta_m)$) are explored.[1]

TopKWY is described in Algorithm 1. In line 1, the threshold δ_m is initialized to α (the threshold with no correction for multiple hypothesis) and σ is initialized accordingly to $\hat{\psi}^{-1}(\delta_m)$. All the elements of the set of minimum p-values $\{p_{\min}^{(j)}\}_{j=1}^{j_p}$ observed on the permuted datasets are initialized to 1 (their maximum achievable value) in line 2. The labels of the permuted datasets are generated in line 3. The pattern exploration is organized using a priority queue Q where each entry represents a pattern S, with key equal to the support x_S and value representing all the information needed by the algorithm regarding S (e.g., a_S) and also with relevant information regarding the parent f_S of pattern S in the search tree (see Section 4). Q is initialized in line 4 and stores the frontier of unexplored patterns, keeping them accessible by non-increasing support. TopKWY stores patterns having p-value $\leqslant \delta_m$ in a priority queue P, keeping them accessible by non-decreasing p-value. This is the set of candidates for $TSP(\mathcal{D}, k, \alpha)$, which are collected and produced in output as soon as possible during the exploration. This allows to reduce

[1] This assumes that the search tree for patterns has the property that the children of a node have support not greater than the node itself, which is a usual property of pattern mining algorithms (Han et al., 2007; Uno et al., 2005; Nijssen and Kok, 2004) and is required by WYlight as well.

the memory requirements and to start analyzing the results during the exploration, without the need of waiting for the algorithm's termination. The first patterns in Q are obtained by the `expand(`S`,`Q`)` operation on line 5 called on the empty pattern $S = \varnothing$: this procedure generates all patterns children of the pattern S in the search tree (and their corresponding projected datasets), and inserts the ones of support $\geqslant \sigma$ in the queue Q. The details of efficient implementations of `expand` are described in Section 6. The `while` loop (lines 6- 21) implements the main step of the exploration strategy: the most frequent pattern S, its support x_S, its support a_S in the minority class of $\mathcal{D}$, and the relevant information for its parent f_S are extracted from Q in line 7. If the p-value p_S of S is $\leqslant \delta_m$, then S is inserted in P in line 10. In line 8, σ' is set to x_S, which is an upper bound to the support of all elements stored in Q. This quantity is used to identify patterns surely in the set $TSP(\mathcal{D}, k, \alpha)$ without waiting for the final corrected significance threshold δ_m to be found, done in lines 11 and 12. k is updated accordingly, reducing it to the number of patterns which still need to be found. In order to compute the corrected significance threshold δ_m, the algorithm computes the p-values of pattern S in the j_p permuted datasets, updating the values of $\{p_{\min}^{(j)}\}_{j=1}^{j_p}$ if needed. This operation is done with the `test` procedure. Similarly to WYlight, our algorithm processes all the j_p permutations for every pattern S at once, computing only the needed exact lower quantile of the set of minimum p-values of the WY permutations, and not the minimum p-values of every permuted dataset. Differently from WYlight, we use an improved lower bound $\psi'(x_S, f_S)$ to the minimum attainable p-value of S to decide (in line 13) whether to test S on the permuted datasets or not (see Section 4). This allows to skip the expensive computation of the j_p supports $\{a_S^{(j)}\}_{j=1}^{j_p}$ of S on the minority class of the permuted datasets for several patterns S.

The significance threshold δ_m is decreased during the exploration in two cases: when the estimated $FWER(\delta_m)$ for the current threshold δ_m increases above α (line 15), or when more than k patterns with p-value $\leqslant \delta_m$ are observed (line 17). The corresponding minimum support threshold σ is then updated accordingly in line 18. The correctness of these steps are proved in Section 3.2. After the update of δ_m and σ, elements which have become untestable are removed from Q in line 19, and elements which are not significant are removed from P in line 20. The current pattern S is expanded in line 21, and all its children having support $\geqslant \sigma$ are inserted into Q. The exploration ends when Q gets empty. When this happens, all elements still contained in P with p-value at most δ_m are reported as significant in line 22.

The strategy employed by TopKWY can be adapted to incrementally update k for the same α, providing an interactive mining process. This can be achieved by providing a maximum value k^* in input to in Algorithm 1 to definitely prune untestable patterns, but freezing the computation after k patterns with p-value below the current value of $\hat{\psi}(\sigma)$ have been found. If the user wants to increase k, the exploration can continue without restarting the entire mining instance.

Algorithm 1: TopKWY

Input: Transaction dataset $\mathcal{D}$ with class labels c, number of permutations j_p, target FWER α, number of results k
Output: Set of top-k significant patterns with FWER $\leqslant \alpha$

1 $\delta_m \leftarrow \alpha$; $\sigma \leftarrow \hat{\psi}^{-1}(\delta_m)$;
2 $p^{(j)}_{\min} \leftarrow 1$, $\forall j \in [1, j_p]$;
3 generate j_p permuted class labels;
4 $Q, P \leftarrow$ empty priority queues;
5 `expand`($\varnothing$, Q);
6 **while** $Q \neq \varnothing$ **do**
7 $(S, x_S, a_S, f_S) \leftarrow Q$.`removeMax`();
8 $\sigma' \leftarrow x_S$;
9 **if** $p_S \leqslant \delta_m$ **then**
10 P.insert(S, p_S);
/* $\mathcal{O} =$ `patterns surely in` $TSP(\mathcal{D}, k, \alpha)$ */
11 $\mathcal{O} \leftarrow \{S' \in P : p_{S'} < \hat{\psi}(\sigma')\}$; produce $\mathcal{O}$ in output;
12 remove patterns in $\mathcal{O}$ from P; $k \leftarrow k - |\mathcal{O}|$;
13 **if** $\psi'(x_S, f_S) \leqslant \delta_m$ **then**
14 `test`(S , $\{p^{(j)}_{\min}\}_{j=1}^{j_p}$) ;
/* `update` δ_m `based on estimate of` δ* */
15 $\delta_m \leftarrow \min\{\delta_m, \max\{\delta : FWER(\delta) \leqslant \alpha\}\}$;
/* `update` δ_m `based on top-`k `patterns in` P */
16 $p^{(k)} \leftarrow k$-th largest p-value in P;
17 $\delta_m \leftarrow \min\{\delta_m, p^{(k)}\}$;
/* `update` σ */
18 $\sigma \leftarrow \psi^{-1}(\delta_m)$;
/* `remove untestable patterns from` Q */
19 remove from Q all patterns S' with $x_{S'} < \sigma$;
/* `remove non-significant patterns from` P */
20 remove from P all patterns S' with $p_{S'} > \delta_m$;
21 `expand`(S , Q);
22 produce in output $\{S' \in P : p_{S'} \leqslant \delta_m\}$;

3.2 Analysis

Some important properties of TopKWY algorithm can be formally stated. The first regards the correctness of the algorithm.

Theorem 1 (Correctness of TopKWY) TopKWY *outputs the set* $TSP(\mathcal{D}, k, \alpha)$ *of top-k significant patterns with* $FWER \leqslant \alpha$.

Proof The correctness of TopKWY follows from two observations: first, the final threshold δ_m obtained by the algorithm is correct; second, only patterns with p-value less or equal than the final value of δ_m are produced in output. We start by proving the first statement. δ_m is initialized to the value α, that is the uncorrected threshold for significance and is always $\geqslant \delta$*. δ_m is decreased (and the corresponding minimum support threshold σ is increased) during the exploration in two cases. The first case (line 15) is when the estimated $FWER(\delta_m)$ for the current threshold δ_m increases above α. This means that

more than αj_p p-values $\{p_{\min}^{(j)}\}_{j=1}^{j_p}$ are below the current significance threshold $\delta_m = \hat{\psi}(\sigma)$, which allows for too many false positives, and the FWER is not correctly controlled to the level α. δ_m is then updated to the highest value of δ for which $FWER(\delta) \leqslant \alpha$. The second case is when more than k patterns with p-value $\leqslant \delta_m$ are observed (line 17). In this case, let $\tilde{p}$ be the highest p-value of the k most significant patterns observed up to this point. Then all patterns of support $< \hat{\psi}^{-1}(\tilde{p})$ cannot result in a p-value $< \tilde{p}$ and therefore we need to consider (both in $\mathcal{D}$ and in the permuted datasets) only patterns of support at least $\hat{\psi}^{-1}(\tilde{p})$. That is, the minimum support threshold σ can be safely increased to $\hat{\psi}^{-1}(\tilde{p})$ with a corresponding significance threshold $\tilde{p}$. When δ_m is last updated, its value will then be equal to the minimum between $\delta(\alpha)$ and $p^{(k)}$.

We now prove the second statement. This is trivially correct for patterns produced in output by line 22. We then consider patterns produced in output in line 11. Note that the current pattern S has support σ' and the search strategy employed by TopKWY guarantees that all patterns with support $> \sigma'$ have already been explored. Therefore, from this point on the algorithm will never encounter p-values $< \hat{\psi}(\sigma')$ and therefore the corrected significance threshold δ_m will be $\geqslant \hat{\psi}(\sigma')$. Thus all patterns in P with p-value $< \hat{\psi}(\sigma')$ can be safely produced in output (and removed from P). □

The following result provides theoretical guarantees on which patterns will be explored by TopKWY, providing analytical evidence of the efficiency of our strategy.

Theorem 2 (Optimality of TopKWY) TopKWY *expands only patterns of support* $\geqslant \hat{\psi}^{-1}(\delta_m)$.

Proof (Proof) Similarly to the proof of Thm. 1, when pattern S of support σ' is extracted from Q, we are guaranteed that the algorithm will never encounter p-values $< \hat{\psi}(\sigma')$ again. Therefore the corrected significance threshold δ_m will be $\geqslant \hat{\psi}(\sigma')$, that is $\sigma' \geqslant \hat{\psi}^{-1}(\delta_m)$ (i.e., S is testable). □

We now show that in a simplified model for how p-values are obtained, there exists a family of datasets for which the expected difference between the number of patterns explored by a DFS strategy and the number of patterns explored by TopKWY is exponential in the size of the dataset. In the simplified model, the p-values obtained by random permutations are uniformly distributed in $(0, 1]$. (Note that we do not assume independence among p-values from different itemsets.) Consider now the family of datasets $\mathcal{D}_n = \{t_1, \ldots, t_n\}$ defined on the set of (binary) features $\mathcal{I} = \{i_1, \ldots, i_n\}$ where $t_j = \mathcal{I} \backslash \{i_j\}$. Moreover, half of transactions in $\mathcal{D}_n$ have label 0 while the other half have label 1.

Theorem 3 *Consider a dataset* $\mathcal{D}_n$ *from the family described above. Let* ε *be a constant such that* $0 < \varepsilon \leqslant \frac{1}{2}$ *and* $\varepsilon n \in \mathbb{N}$. *Assume that the choice of* α *and* k *is such* $\delta^* = \hat{\psi}(\varepsilon n) = \binom{\frac{n}{2}}{\varepsilon n} / \binom{n}{\varepsilon n}$, *and that* j_p *random permutations are used.*

Let X be the difference between the number of patterns explored by a DFS strategy and the number of patterns explored by TopKWY *in the simplified model above. Then $\mathbb{E}[X] = \Omega\left(2^{\varepsilon n}/j_p\right)$.*

Proof Note that in such dataset all patterns are closed. Let W be the set of patterns explored by TopKWY. The DFS strategy has to explore all the patterns in W (since they are testable). We now show that it will also explore, in expectation, $\Omega\left(2^{\varepsilon n}/j_p\right)$ additional patterns, that proves the statement.

It is easy to show that since $\varepsilon n \in \mathbb{N}$ and $\delta^* = \binom{\frac{n}{2}}{\varepsilon n}/\binom{n}{\varepsilon n}$, the set of testable patterns W is given by all patterns of size $\leqslant \varepsilon n$. For each pattern S_i and each j with $1 \leqslant j \leqslant j_p$, let X_{ij} be the random variable that is 1 if S_i has p-value $\leqslant \delta^*$ in the j-th permuted dataset, and 0 otherwise. Note that S_{ij} is a Bernoulli random variable of parameter δ^*, for all i and j. Let Y be the number of p-values lower than δ^*, assuming that the DFS has explored $\varepsilon n + m$ patterns. The expectation of Y is

$$\mathbb{E}[Y] = \mathbb{E}\left[\sum_{i=1}^{\varepsilon n+m}\sum_{j=1}^{j_p} X_{ij}\right] = \sum_{i=1}^{\varepsilon n+m}\sum_{j=1}^{j_p}\mathbb{E}[X_{ij}] = (\varepsilon n + m) j_p \delta^*. \tag{3}$$

Note that requiring $\mathbb{E}[Y] \geqslant 1$ provides a lower bound to the number of p-values needed for the DFS to establish that $\delta^* = \binom{\frac{n}{2}}{\varepsilon n}/\binom{n}{\varepsilon n}$, since αj_p p-values below such threshold must be observed.

Let $v = \varepsilon n$. The condition $\mathbb{E}[Y] = (\varepsilon n + m) j_p \delta^* \geqslant 1$ implies

$$\begin{aligned}
(\varepsilon n + m) &\geqslant \frac{1}{j_p \delta^*} = \frac{1}{j_p}\frac{\binom{n}{v}}{\binom{\frac{n}{2}}{v}} = \frac{1}{j_p}\frac{n!(\frac{n}{2}-v)!}{(\frac{n}{2})!(n-v)!} \\
&= \frac{1}{j_p}\frac{(n-v)!\prod_{j=0}^{v-1}(n-j)}{(n-v)!}\frac{(\frac{n}{2}-v)!}{(\frac{n}{2}-v)!\prod_{j=0}^{v-1}(\frac{n}{2}-j)} \\
&= \frac{1}{j_p}\prod_{j=0}^{v-1}\frac{(n-j)}{(\frac{n}{2}-j)} \geqslant \frac{1}{j_p}\prod_{j=0}^{v-1} 2 = \frac{2^v}{j_p} = \frac{2^{\varepsilon n}}{j_p}
\end{aligned}$$

or, equivalently

$$m \geqslant \frac{2^{\varepsilon n}}{j_p} - \varepsilon n \in \Omega\left(2^{\varepsilon n}/j_p\right).$$

Note that since the set of testable patterns includes all patterns of size $\leqslant \varepsilon n$, all the $\Omega\left(2^{\varepsilon n}/j_p\right)$ patterns explored by the DFS *after* exploring the first εn patterns and before observing the first p-value below δ^* have size $> \varepsilon n$ and, thus, are not in W, which proves the statement.

□

Compared to a depth first search (DFS) exploration strategy (i.e., the one employed by WYlight), the *best first* exploration strategy followed by TopKWY has the additional costs required by operations involving data structures P and Q. P can be implemented as a heap of entries (p, ℓ_p), where the key p is a p-value and the value ℓ_p is a list of patterns which contains all the patterns

with the same p-value p. With such implementation, operations involving P (lines 10, 11, 16, and 20) can be perfomed with $O(\log k)$ operations, since P will contain at most k entries. Analogously, Q can be implemented as a heap of entries (x, ℓ_x), where the key x is a support and the value ℓ_x is a list of patterns with the same value of $\hat{\psi}(x)$ (that is, having the same support x). With such implementation, operations involving Q (lines 7, 19, and 21) can be performed with $O(\log n)$ operations (n is the number of transactions in $\mathcal{D}$). Alternatively, P and Q can be implemented so that all operations require time $O(1)$ by storing references to lists ℓ_p and ℓ_x in arrays of size $O(n^2)$ (since the p-value of S is a function of the support x_S of S and the number a_S of transactions with label 1 and containing S) and $O(n)$, respectively. (Note that this additional space requirement is not impractical, since $O(nj_p)$ space is needed to store the random permutations.) We also note that computing the improved lower bound $\psi'(x_S, f_S)$ (line 13), has the same cost as computing the lower bound $\hat{\psi}(x_S)$ to the p-value used in WYlight (see Section 4). Even with the additional costs required by P and Q, the best first strategy of TopKWY leads to significant improvements in running time, as demonstrate by our experimental evaluation (Section 7).

4 Improved Bounds on Minimum Attainable p-value

In this section we prove novel and efficiently computable lower bounds on the minimum p-value achievable by a pattern S that are tighter than the ones introduced by LAMP (Terada et al., 2013a) and are of particular interest in the context of WY permutation testing. These bounds are based on information computed when processing a *parent pattern* Y of S; in the case of itemsets, Y is a parent of S (or, alternatively, S is a *child* or a *super pattern* of Y) when $Y \subset S$. Such bounds can be used to *skip* the expensive processing of the permutations for S when they ensure that it is not possible to improve the current estimate of the corrected significance threshold. While we present these bounds as a critical component of TopKWY, they may be of independent interest since can be employed in WYlight or similar algorithms to speed-up WY permutation testing.

Let the pattern S be a super pattern of Y, that is $S \supset Y$. Then $x_Y \geqslant a_Y \geqslant 0$ and $x_Y \geqslant x_S$. Since the set of transactions (i.e., the *conditional dataset*) containing S is a subset of the set of transactions containing Y, we can bound the support a_S of S in the class c_1 with the following relations:

$$\max(a_Y - (x_Y - x_S), 0) \leqslant a_S \leqslant \min(x_S, a_Y).$$

Considering the j_p permuted class labels, let $a_Y^{(j)}$ be the number of transactions containing Y and in the minority class (i.e., $a_Y^{(j)}$ is the value of a_Y when the class labels are given by the j-th permutation).An analogous relation holds between $a_S^{(j)}$ and $a_Y^{(j)}$, for all j:

$$\max(a_Y^{(j)} - (x_Y - x_S), 0) \leqslant a_S^{(j)} \leqslant \min(x_S, a_Y^{(j)}).$$

An immediate consequence of these bounds on $a_S^{(j)}$ are *lower bounds* to $p_S(a_S^{(j)})$.

Lemma 2 *Let* $\breve{a}_S^{(j)} = \max(a_Y^{(j)} - (x_Y - x_S), 0)$ *and* $\hat{a}_S^{(j)} = \min(x_S, a_Y^{(j)})$. *Then, for all* $j \in [1, j_p]$,

$$p_S\left(a_S^{(j)}\right) \geqslant \min\left\{p_S\left(\breve{a}_S^{(j)}\right), p_S\left(\hat{a}_S^{(j)}\right)\right\}.$$

This result suggests that if we have already computed $a_Y^{(j)}, \forall j \in [1, j_p]$ while processing the permutations of Y, we could skip the expensive computation of $a_S^{(j)}$, and, therefore, $p_S(a_S^{(j)}), \forall j \in [1, j_p]$, in situations when the lower bounds to $p_S(a_S^{(j)})$ are *greater* than the current value of the corrected significance threshold. In the following, we present a bound valid for all $p_S(a_S^{(j)})$ simultaneously, that is a function of only the minimum and maximum elements of $a_Y^{(j)}$, instead of all of them. Let

$$a_{Y_{\min}} = \min\left\{a_Y^{(j)} : j \in [1, j_p]\right\}$$

and

$$a_{Y_{\max}} = \max\left\{a_Y^{(j)} : j \in [1, j_p]\right\} \quad .$$

Then, $\forall j \in [1, j_p]$ we bound $a_S^{(j)}$ as:

$$a_{S_{\min}} = \max(a_{Y_{\min}} - (x_Y - x_S), 0) \leqslant a_S^{(j)} \leqslant \min(x_S, a_{Y_{\max}}) = a_{S_{\max}} \quad .$$

This allows to compute a bound $\psi'(x_S, x_Y, a_{Y_{\min}}, a_{Y_{\max}})$ to the minimum attainable p-value of S that is tighter than $\psi(x_S)$:

$$\psi'(x_S, x_Y, a_{Y_{\min}}, a_{Y_{\max}}) = \min(p_S(a_{S_{\min}}), p_S(a_{S_{\max}})) \quad . \tag{4}$$

The bound in Equation 4 is evaluated in constant time, assuming $p_S(a)$ is pre-computed for all valid values of a, as done in WYlight and in TopKWY to efficiently implement the `test` function. The following are simple consequences of Lemma 2 and the fact that $a_{S_{\min}}$ and $a_{S_{\max}}$ are always equally or more tight than the naive bounds on a_S assumed by $\psi(x_S)$.

Lemma 3 $\min\left\{p_S\left(a_S^{(j)}\right) : j \in [1, j_p]\right\} \geqslant \psi'(x_S, x_Y, a_{Y_{\min}}, a_{Y_{\max}}) \geqslant \psi(x_S)$.

If for the current value of the significance threshold δ_m it holds that $\psi'_S > \delta_m$, then we can infer, without computing $\{a_S^{(j)}\}_{j=1}^{j_p}$, that none of the j_p p-values of S in the permuted datasets will improve the estimate of the current lower-quantile of the set $\{p_{\min}^{(j)}\}_{j=1}^{j_p}$ and therefore cannot contribute to the computation of $\delta(\alpha)$ or δ_m. That is, all the computation on the permuted datasets can be skipped for the current pattern S. For all children of S, if S is not tested the bounds $a_{S_{\min}}$ and $a_{S_{\max}}$ can be propagated to compute bounds also on their class distribution; if S is tested, then we propagate the actual

minimum and maximum values of $\{a_S^{(j)}\}_{j=1}^{j_p}$. In Algorithm 1 we use the bound above with the values propagated by the parent f_S of S and use $\psi'(x_S, f_S)$ to highlight this fact. This optimization is particularly effective when patterns have a high degree of correlation, i.e., when patterns share many transactions.

Note that even if S does not need to be tested, descendants of S may need to be tested. However, using the bound $\psi'(\cdot)$ we can quickly identify cases in which none of the descendants of S need to be explored and therefore the entire subtree can be pruned. In particular, since all the descendants of S will have support $\leqslant x_S - 1$, considering a_S (i.e., the number of transactions containing S and in the minority class in the dataset $\mathcal{D}$), the algorithm can find $\min\{\psi'(i, x_S, a_{S_{\min}}, a_{S_{\max}}) : i \in [\sigma, x_s - 1]\}$, and if such value is $> \delta_m$ we can prune all the search subtrees rooted in the children of S. This optimization is part of the `expand` operation in TopKWY. These novel bounds consider the information of one common ancestor pattern to avoid useless computations for many of its children: in practice, the number of tests to perform across the permuted datasets can be significantly smaller than the number of testable patterns, leading to a significant computational speed-up. The approach above can extended to bound $\min\left\{p_S\left(a_S^{(j)}\right) : j \in [1, j_p]\right\}$ by considering the information computed on the intersection of the conditional datasets of any pair of patterns S and Y, even if $S \pitchfork Y$.

Another possible extension of the techniques we derive in this section is to consider not only *one pair* $(a_{Y_{\min}}, a_{Y_{\max}})$, but v *pairs* $(a^i_{Y_{\min}}, a^i_{Y_{\max}})$, for all $i \in [1, v]$: for each i, we define the set J_i as a subset of $\{1, \ldots, j_p\}$, with $\bigcup_{i=1}^{v} J_i = \{1, \ldots, j_p\}$. For all i, we bound all values $a_Y^{(j)}$ for all $j \in J_i$ with $a^i_{Y_{\min}}$ and $a^i_{Y_{\max}}$. If we define

$$a^i_{Y_{\min}} = \min\left\{a_Y^{(j)} : j \in J_i\right\} \quad , \quad a^i_{Y_{\max}} = \max\left\{a_Y^{(j)} : j \in J_i\right\},$$

$$a^i_{S_{\min}} = \max(a^i_{Y_{\min}} - (x_Y - x_S), 0) \quad , \quad a^i_{S_{\max}} = \min(x_S, a^i_{Y_{\max}}),$$

then we simply obtain

$$\min\left\{p_S\left(a_S^{(j)}\right) : j \in J_i\right\} \leqslant \min\left\{p_S\left(a^i_{S_{\min}}\right), p_S\left(a^i_{S_{\max}}\right)\right\}.$$

This provides a trade-off between the memory (required to store the $2v$ bounds on the expanded nodes of the search space) and the time to evaluate the bound (that is linear in the number v of sets instead of constant), and the time the algorithm saves by skipping computations of the permutations, that is a direct consequence of how tight the bounds on the p-values are; in fact, the permutations that have to be processed are only the ones belonging to the sets J_i such that $\min\left\{p_S\left(a^i_{S_{\min}}\right), p_S\left(a^i_{S_{\max}}\right)\right\}$ is not higher than the current value of the significance threshold.

5 Extensions of TopKWY

5.1 Controlling the Generalized FWER

While the main focus of TopKWY is to control the FWER of the output, a simple modification provides an algorithm to control the *generalized* FWER (g-FWER (Lehmann and Romano, 2012)). The g-FWER is defined as the probability that at least g false positives are reported in output. In several applications one may be willing to tolerate a small amount of false discoveries in order to increase the power of detecting significant patterns, provided the number of false discoveries can be controlled. In such cases methods to discover significant patterns while controlling the g-FWER are preferred to methods controlling FWER.

Let FP be the number of false positives, then: g-FWER$= \Pr(FP \geqslant g)$. Let $g\text{-}FWER(\delta)$ be the g-FWER obtained using δ as *corrected* significance threshold, that is by flagging as significant patterns with p-value $\leqslant \delta$. Note that the Westfall-Young procedure can be used to estimate $g\text{-}FWER(\delta)$ as

$$g\text{-}FWER(\delta) = \frac{1}{j_p} \sum_{j=1}^{j_p} \mathbb{1}[p_g^{(j)} \leqslant \delta]$$

where $p_g^{(j)}$ is the g-th smallest p-value (over all patterns) in the permuted dataset j.

Algorithm 1 can be simply modified to obtain the set of top-k significant patterns with g-FWER $\leqslant \alpha$. To achieve this, it is sufficient to perform the following changes: replace $\texttt{test}(S\ ,\ \{p_{\min}^{(j)}\}_{j=1}^{j_p})$ (line 14) with $\texttt{test}(S\ ,\ \{p_g^{(j)}\}_{j=1}^{j_p})$, where $\texttt{test}(S\ ,\ \{p_g^{(j)}\}_{j=1}^{j_p})$ computes the p-value of pattern S in the j_p permuted datasets and updates the values of $\{p_g^{(j)}\}_{j=1}^{j_p}$ if needed; replace $\max\{\delta : FWER(\delta) \leqslant \alpha\}$ (line 15) with $\max\{\delta : g\text{-}FWER(\delta) \leqslant \alpha\}$. Let TopKWY-$g$ be such modified algorithm. We have the following.

Lemma 4 TopKWY-g *outputs the set* $TSP(\mathcal{D}, k, \alpha)$ *of top-*k *significant patterns with* $g\text{-}FWER \leqslant \alpha$.

The proof is analogous to the proof of Theorem 1.

In addition to finding the top-k most significant pattern with bounded g-$FWER$, with g provided in input, TopKWY-g can be adapted to a different scenario. In this case, one may want to retrieve the k most significant patterns, using the random permutations to obtain a rigorous *estimate* of how many of such results are likely to be false positives. More formally, for k and α provided by the user, one may be interested in computing the quantity $g^\star$ defined as

$$g^\star = \min\left\{g : g\text{-}FWER(p^k) \leqslant \alpha\right\},$$

that is the *mininum* value of g such that the g-$FWER$ is controlled when the significance threshold is $p^{(k)}$, that is the highest p-value of the set of top-k

results we extracted. $g^\star$ provides useful knowledge on the *quality* of the set of top-k results provided to the user. In this situation the user is not required to fix a-priori g before examining the data. Further simple modifications to TopKWY-g are sufficient to obtain such variant, that are: remove line 15 and replace line 22 with "produce in output $\{S' \in P : p_{S'} \leqslant \delta_m\}$ and $g^\star = \min\left\{g : g\text{-}FWER(p^k) \leqslant \alpha\right\}$". Let TopKWY* be such modified algorithm; we obtain the following guarantees.

Lemma 5 TopKWY* *outputs the set of patterns* $\left\{S : p_S \leqslant p^k\right\}$ *of top-*k *significant patterns and* $g^\star = \min\left\{g : g\text{-}FWER(p^k) \leqslant \alpha\right\}$.

5.2 Bounding the Proportion of False Discoveries

The False Discovery Proportion (FDP) (van der Laan et al., 2004; Lehmann and Romano, 2012) of a set of hypotheses $\mathcal{P}$ is defined as the ratio $FD/|\mathcal{P}|$, where FD is the (unknown) number of false discoveries $\in \mathcal{P}$; note that when $|\mathcal{P}| = 0$, the FDP is assumed to be 0. Let $\zeta, \alpha \in (0,1)$ and $k, g \in [1, +\infty)$. Define the set $\mathcal{P}$ such that all the following hold:

$$\begin{aligned} &\max\{p_S : (S, p_S) \in \mathcal{P}\} \leqslant \delta^*, \\ &\delta^* = \max\{\delta : g\text{-}FWER(\delta) \leqslant \alpha\}, \\ &g \leqslant \zeta|\mathcal{P}|, \quad |\mathcal{P}| \leqslant k. \end{aligned}$$

It is possible to prove that a set $\mathcal{P}$ satisfying the above guarantees has size at most k and $FDP \leqslant \zeta$ with probability $\geqslant 1 - \alpha$. Simple modifications to TopKWY-g lead to an algorithm that outputs $\mathcal{P}$ with the aforementioned guarantees: remove lines 11 and 12, replace line 15 with "$\delta_m \leftarrow \max\{\delta : (\lfloor k\zeta \rfloor)\text{-}FWER(\delta) \leqslant \alpha\}$" and line 22 with "produce in output $\mathcal{P}(\delta^*) = \{S' \in P : p_{S'} \leqslant \delta^*\}$ where $\delta^* = \max\{\delta : (\lfloor \zeta|\mathcal{P}(\delta)| \rfloor)\text{-}FWER(\delta) \leqslant \alpha\}$". Let such algorithm be TopKWY-ζ. We obtain the following result.

Lemma 6 TopKWY-ζ *outputs the set of patterns* $\mathcal{P}$ *of size at most* k *with False Discovery Proportion* $\leqslant \zeta$ *with probability* $\geqslant 1 - \alpha$.

5.3 Alternative Exploration Strategies

While TopKWY builds on examining the search tree of all possible patterns in order of decreasing support, i.e. with a *best first strategy* analogous to the one used by TopKMiner (Pietracaprina and Vandin, 2007) for mining top-k frequent patterns, it can also be modified to efficiently obtain the set $TSP(\mathcal{D}, k, \alpha)$ using different exploration strategies, e.g. a level-wise exploration of the search tree (performed, e.g., by the Apriori algorithm (Agrawal and Srikant, 1994) for itemsets) or a depth first search on the tree of all possible patterns (Uno et al., 2005; Nijssen and Kok, 2004). This can be achieved by setting σ' to $\max\{x_{S'} : S' \in Q\}$ (instead that to x_S) in line 8 of Alg. 1 and by

an appropriate choice of the priority for patterns in the priority queue Q, that stores the frontier of unexplored patterns: to obtain a level-wise exploration for itemsets, the priority of pattern S is set to the total number $|\mathcal{I}|$ of items minus $|S|$; to obtain a depth first search, the priority of patterns S is set to its level in the search tree.

While for strategies other then the *best first* one the optimality (Theorem 2) is not guaranteed, the possibility to employ other strategies allows to obtain the top-k significant patterns starting from efficient implementations of frequent pattern mining algorithms that build on such strategies for various types of patterns (e.g., subgraphs (Nijssen and Kok, 2004)).

6 Implementation Details

An efficient implementation of *expand* and *test* procedures is critical for the efficiency of TopKWY. This crucially depends on the representation of $\mathcal{D}$ and the permuted class labels, and both depend on the type of patterns of interest. In Sections 6.1 and 6.2 we now describe in more details the implementations for significant itemsets mining and for significant subgraphs mining; in particular, for significant itemsets we discuss the implementation of TopKWY as described in Section 3.1 as well as the variant, described in Section 5.3 of TopKWY based on the DFS strategy, which we denote by TopKWY-dfs; for significant subgraphs we only consider the implementation of TopKWY-dfs.

6.1 Significant Itemset Mining

Our implementation of TopKWY is based upon TopKMiner (Pietracaprina and Vandin, 2007), which mines top-k frequent closed itemsets. As for TopKMiner, TopKWY uses a PatriciaTrie (Zandolin and Pietracaprina, 2003) to store a compact representation of the dataset $\mathcal{D}$ in which transactions sharing the same prefix are represented by the same node in the tree. The conditional dataset of (i.e., the set transactions containing) an itemset Y is stored as a list m_Y of nodes of the PatriciaTrie. An additional counter is added to every node, representing how many transactions with prefix represented by the node belong to the class 1. The same is done for the j_p permutations adding j_p counters to each node in the trie. Since the PatriciaTrie is built adding one transaction at a time, a technique similar to reservoir sampling is used to generate the j_p permuted labels of every transaction. Let $\mathbf{r}$ be a vector of length j_p, with all components $r^{(j)}, j = 1, \ldots, j_p$ of $\mathbf{r}$ initialized to n_1, the number of transactions to assign to the class c_1 for every permutation of index j. For every transaction t_i, with $i \in [1, n]$, the j-th label of t_i is assigned to c_1 with probability $r^{(j)}/(n - i + 1)$, c_0 otherwise. If c_1 is chosen, then $r^{(j)}$ is decreased of one. This method guarantees that the total number of transactions with label c_1 will be n_1 for every $j \in [1, j_p]$ and that the labels of the j-th permuted dataset are obtained by a random permutation of the class lables.

Our implementation of TopKWY-dfs is based upon LCM 3 (Uno et al., 2005), which mines frequent closed itemsets using a depth first strategy. The third version of LCM combines various techniques and data structures to accelerate the generation and the computation of the frequencies of frequent closed itemsets.

6.2 Significant Subgraph Mining

Our implementation of TopKWY-dfs relies on Gaston (Nijssen and Kok, 2004) to mine significant subgraphs. Gaston first considers simple patterns, such as paths and trees, since efficient techniques for isomorphism checking are available for such acyclic structures. Only after this first phase, denoted as "quickstart", general subgraphs, containing cycles, are evaluated. The search strategy of Gaston relies on a depth first enumeration of subgraphs. We do not provide a subgraph variant of TopKWY because no competitive algorithms based on a best first exploration strategy are currently available (Wörlein et al., 2005; Nijssen and Kok, 2006).

7 Experimental Evaluation

We implemented and tested TopKWY and TopKWY-dfs for the extraction of significant itemsets and significant subgraphs. Our experimental evaluation has three goals. First, to assess the number of significant patterns found in real datasets. Second, to evaluate the performance of TopKWY: since no other tool for the extraction of top-k significant patterns exists, we compare TopKWY and TopKWY-dfs with the state-of-the-art tool for significant pattern mining, WYlight (Llinares-López et al., 2015). While the techniques introduced in this work can be extended to other multiple hypothesis testing procedures, such as LAMP (Terada et al., 2013a), we do not compare with LAMP or derived strategies (Minato et al., 2014) since (Llinares-López et al., 2015) shows that WY permutation testing results in higher power. Third, to assess the impact of our improved bounds and implementation choices on performances.

In Section 7.1 we describe the implementation and computational environment for our experiments. In Section 7.2 we describe the datasets we used. In Section 7.3 we describe the experiments we have performed and our choice of parameters. Finally, in Section 7.4 we report and discuss the results of our experiments.

7.1 Implementation and Environment

We implemented TopKWY in C++ as an extension of the TopKMiner algorithm (Pietracaprina and Vandin, 2007). For TopKWY-dfs we modified the C implementation of WYlight (based on LCM (Uno et al., 2005) and

Table 1 Itemset Datasets statistics. For each dataset the table reports: the number $|D|$ of transactions; the number $|I|$ of items; the average transaction length avg; the fraction n_1/n of transactions in the minority class; the number $SP(0.05)$ of significant patterns for $FWER = 0.05$.

dataset	$\|D\|$	$\|I\|$	avg	n_1/n	$SP(0.05)$
svmguide3(L)	1,243	44	21.9	0.23	36,736
chess(U)	3,196	75	37	0.05	$> 10^7$
mushroom(L)	8,124	118	22	0.48	71,945
phishing(L)	11,055	813	43	0.44	$> 10^7$
breast cancer(L)	12,773	1,129	6.7	0.09	6
a9a(L)	32,561	247	13.9	0.24	348,611
pumb-star(U)	49,046	7117	50.5	0.44	$> 10^7$
bms-web1(U)	58,136	60,978	2.51	0.03	704,685
connect(U)	67,557	129	43	0.49	$> 10^8$
bms-web2(U)	77,158	330,285	4.59	0.04	289,012
retail(U)	88,162	16,470	10.3	0.47	3,071
ijcnn1(L)	91,701	44	13	0.10	607,373
T10I4D100K(U)	100,000	870	10.1	0.08	3,819
T40I10D100K(U)	100,000	942	39.6	0.28	5,986,439
codrna(L)	271,617	16	8	0.33	4,088
accidents(U)	340,183	467	33.8	0.49	$> 10^7$
bms-pos(U)	515,597	1,656	6.5	0.40	26,366,131
covtype(L)	581,012	64	11.9	0.49	542,365
susy(U)	5,000,000	190	43	0.48	$> 10^7$

Gaston (Nijssen and Kok, 2004)) made available by the authors at `https://github.com/fllinares/wylight`. All implementations were compiled with `gcc 4.8.4`. Our experiments have been performed on a 2.30 GHz Intel Xeon CPU machine with 512 GB of RAM, running on Ubuntu 14.04. Our code and scripts to replicate all experiments described in the paper are available at `https://github.com/VandinLab/TopKWY`.

7.2 Datasets

For itemsets mining, we performed our experiments using 19 datasets: the 10 largest ones used in Llinares-López et al. (2015) and available at FIMI'04[2] and UCI[3], all the datasets used in Komiyama et al. (2017), available from the libSVM repository[4], and 4 additional ones (a9a, bms-web1, accidents, susy) available from libSVM, FIMI'04, and SPMF[5]. The datasets' statistics are in Table 1. For each dataset, we also note if it already contained class labels (L) or not (U). For unlabeled datasets we simulated a typical analysis requiring

[2] `http://fimi.ua.ac.be`

[3] `https://archive.ics.uci.edu/ml/index.php`

[4] `https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/`

[5] `http://www.philippe-fournier-viger.com/spmf`

Table 2 Subgraph Datasets statistics. For each dataset the table reports: the number $|D|$ of graphs; the average number of nodes $|V_{avg}|$; the average number of edges $|E_{avg}|$; the fraction n_1/n of graphs in the minority class; the number $SP(0.05)$ of significant patterns for $FWER = 0.05$. The number in brackets report the maximum number of vertexes of the explored subgraphs, that has been limited to allow practical running times for the experiments.

dataset	$\|D\|$	V_{avg}	E_{avg}	n_1/n	$SP(0.05)$
MUTAG	188	17.93	19.79	19.79	70,184
BZR(30)	405	35.75	38.36	0.21	80,425
COX2	467	41.22	43.45	0.22	$> 10^6$
ENZYMES(10)	600	32.63	62.14	0.17	112,158
DHFR(30)	753	42.43	44.54	0.61	$> 10^6$
DD	1,178	284.32	715.66	0.58	80,256
AIDS(30)	2,000	15.69	16.20	0.2	566,727
NCI1	4,110	29.87	32.30	0.5	$> 10^6$
NCI109	4,127	29.68	32.13	0.5	$> 10^6$
Mutagenicity	4,337	30.32	30.77	0.44	$> 10^6$
Tox_21_AHR(30)	8,169	18.09	18.50	0.12	98,398

to find itemsets correlated with a given item (feature) in a dataset. For every unlabeled dataset we selected the single item whose frequency is closer from below to 0.5, removed the corresponding item from every transaction, and use its appearance to define the target class label. The reported ratio n_1/n for the minority class of unlabeled datasets refers to the output of this labeling process. For real-valued features we obtained two bins by thresholding at the mean value and using one item for each bin (analogously to Komiyama et al. (2017)).

For subgraphs mining, we considered 11 of the largest datasets with binary target labels available from a repository[6] of benchmark datasets; most of them are also analysed by Llinares-López et al. (2015). The datasets' statistics are in Table 2. In some cases, we bounded the maximum number of vertexes of the subgraphs that are explored by Gaston, in order to obtain practical running times for the experiments. Such limits are reported in the parenthesis after the dataset's name in Table 2.

7.3 Parameters and Experiments

For TopKWY and TopKWY-dfs we considered $k = 10^i$ for $i \in [1, 6]$. For all the datasets we analyzed, we ran TopKWY and TopKWY-dfs, for all such values of k, and WYlight. We fixed the number of permutations $j_p = 10^4$, shown to be a good choice in Llinares-López et al. (2015), and fixed the commonly used value $\alpha = 0.05$ as FWER threshold. For the comparison between TopKWY, TopKWY-dfs, and WYlight, we repeated every experiment 10

[6] `https://ls11-www.cs.tu-dortmund.de/staff/morris/graphkerneldatasets`

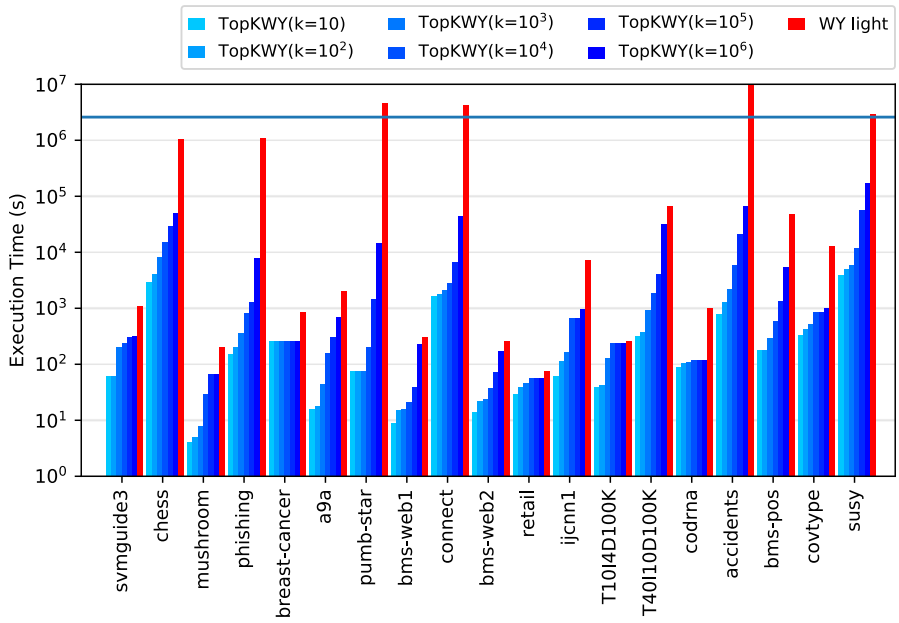

Fig. 2 Running time for TopKWY (with various values of k) and WYlight. The blue horizontal line corresponds to 1 month of computation.

times, recording the running time and peak memory provided by the operating system; we report the averages over the 10 runs, standard deviations are negligible and therefore not shown. The measures reported for TopKWY include the time and space to retrieve statistically significant patterns and write them on file, while for TopKWY-dfs and WYlight we only report the time and space needed to find the optimal significance threshold, which corresponds to the first step of the method, therefore reporting a lower bound to their runtimes. We stopped the execution of an algorithm if it did not conclude after (at least) one month of computation; for these cases, the indicated time and peak memory are lower bounds. For experiments testing the impact of parameters or implementation choices on TopKWY we used only one execution.

7.4 Results

7.4.1 Itemsets Mining

Table 1 reports the number of significant patterns for $\alpha = 0.05$ in the datasets we considered, obtained by running TopKWY (with $k = +\infty$) or WYlight. For some datasets we stopped the computation after 1 month, so only a lower bound is available. In most cases, the number of significant patterns is extremely large: for 11 out of 19 datasets there are $> 5 \times 10^5$ significant patterns and in 7 datasets there are $> 10^7$ significant patterns. Therefore a direct way

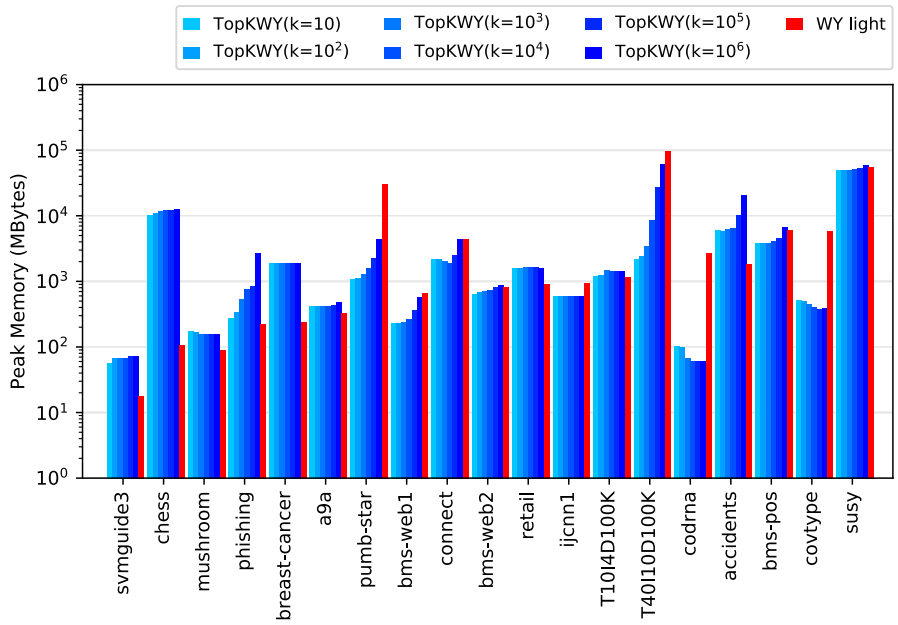

Fig. 3 Peak memory for TopKWY (with various values of k) and WYlight.

to limit the number of significant patterns in output, as provided by the top-k significant patterns, is required.

Figure 2 compares the running time of TopKWY and WYlight. Note that for the 11 datasets in which the number of significant patterns is $< 10^6$, TopKWY with $k = 10^6$ identifies all the significant patterns and produces the same patterns found with WYlight. For 15 out of 19 datasets, TopKWY (with $k = 10^6$) is faster than WYlight by a factor at least 2. For 9 datasets TopKWY is faster than WYlight by at least one order of magnitude, and for 6 datasets WYlight requires > 11 days while TopKWY identifies up to the 10^6 most significant patterns within one day and the 10^4 most significant ones in few hours. Even for the datasets where TopKWY identifies all significant patterns, producing the same patterns as WYlight, TopKWY is always faster than WYlight, with up to one order of magnitude speed-up in some cases. This shows that TopKWY is an effective tool to identify all significant patterns whenever possible and enables the analysis of significant patterns when their number is extremely high. For datasets in which the number of significant patterns is $> 10^6$ we ran TopKWY with $k = \infty$ to compare its strategy for finding the corrected significance threshold for *all* significant patterns with the one used by WYlight. The runtime of TopKWY is always lower than the runtime of WYlight by at least 20%, with a significant speed-up in some case (e.g., for chess, TopKWY terminates in 2 days, while WYlight needs more than 10 days). These results show that TopKWY outperform the state-of-the-art even for this task.

Figure 3 compares the peak memory required by TopKWY and WYlight. Given the *best first* strategy employed by TopKWY, we expected its memory requirement could be higher than WYlight, that follows a depth first strategy. Interestingly, only in three cases TopKWY required 1 order of magnitude more memory than WYlight and in both such cases the requirements are reasonable ($\leqslant$ 20 GBs) for current machines. However, in such cases WYlight required $>$ 11 days to complete, while TopKWY terminated in $<$ 1 day, showing that, by using a reasonably larger amount of memory than WYlight, TopKWY renders the identification of significant patterns feasible. In all other cases the memory requirement of TopKWY is either the same or within few GBs of WYlight. For some datasets TopKWY requires significantly less memory than WYlight: surprisingly this happens for datasets (cod-rna, covtypes) on which TopKWY reports the same significant patterns as WYlight (i.e., *all* significant patterns). In some cases, memory usage decreases slightly when k increases, due to our dynamical allocation of the p-values lookup table that may require less space when the minimum support decreases.

We investigated the impact of our implementation choices on the memory requirement of TopKWY (Figure 4). We compared the space required to store the permuted labels on all the nodes of the PatriciaTrie used by TopKWY (see Section 6.1) with the space required by storing the permuted labels for each transaction (as done for example by WYlight). Since TopKWY stores, for each node of the Patricia Trie, a list of j_p values (i.e., the number of transactions with minority label among the ones sharing the prefix corresponding to the node), one transaction may have more than j_p values associated to its nodes. In most cases the space required by the two methods is essentially the same, but in three cases the use of the Patricia Trie corresponds to a significant reduction in the memory used. In particular, these three cases are for datasets in which TopKWY identifies all the significant patterns using less memory than WYlight, providing strong evidence of the importance of our encoding of the permuted class labels.

We compared the exploration strategies used by TopKWY and by WYlight by recording the number of patterns they test (Figure 5), restricting to datasets in which WYlight terminates. In all cases, TopKWY tests a lower number of patterns than WYlight, with differences of almost two orders of magnitude for some datasets. This shows the effectiveness of our exploration strategy and of our novel bounds $\psi'(\cdot)$ (see Section 4) on reducing the number of tests to perform.

We then directly investigated the impact of our novel bounds on the runtime of TopKWY. We compared the running time of WYlight with the running time of two variants of TopKWY: one using our improved bound $\psi'(\cdot)$ and one using the LAMP bound $\hat{\psi}(\cdot)$ (i.e., the same bound used by WYlight). The results for some representative datasets are in Figure 6(a). The results for the other datasets are similar. We observed that, for all datasets other than chess, the exploration strategy employed by TopKWY to extract only the top-k significant patterns already provides a substantial (up to more than one order of magnitude) improvement in the running time of TopKWY with

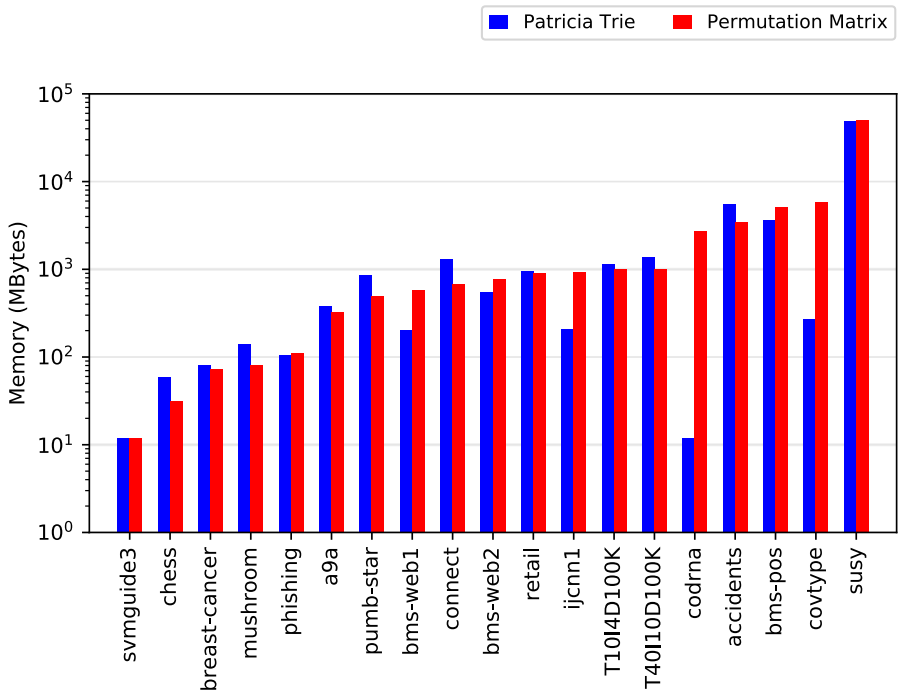

Fig. 4 Memory requirement for permuted class labels using PatriciaTrie and permutation matrix.

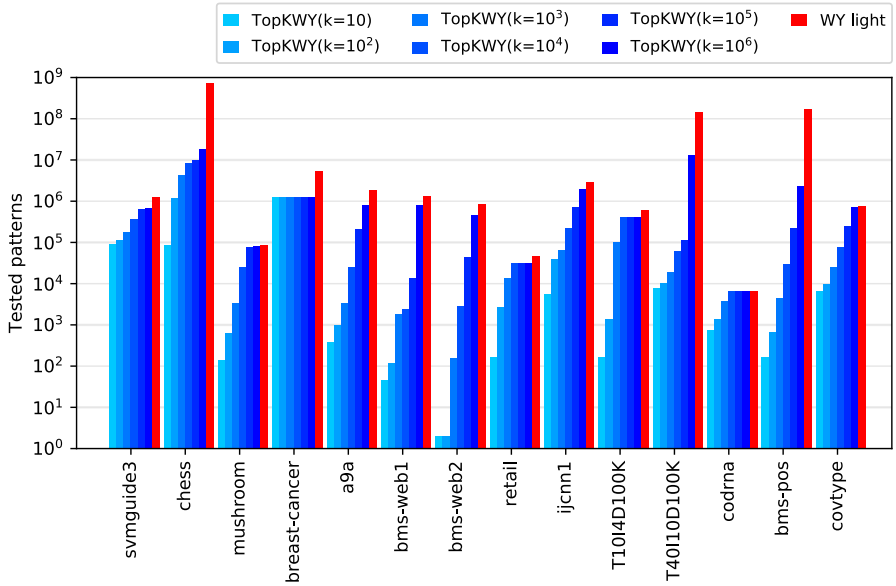

Fig. 5 Comparison between the number of tested patterns on the permuted datasets by TopKWY and WYlight.

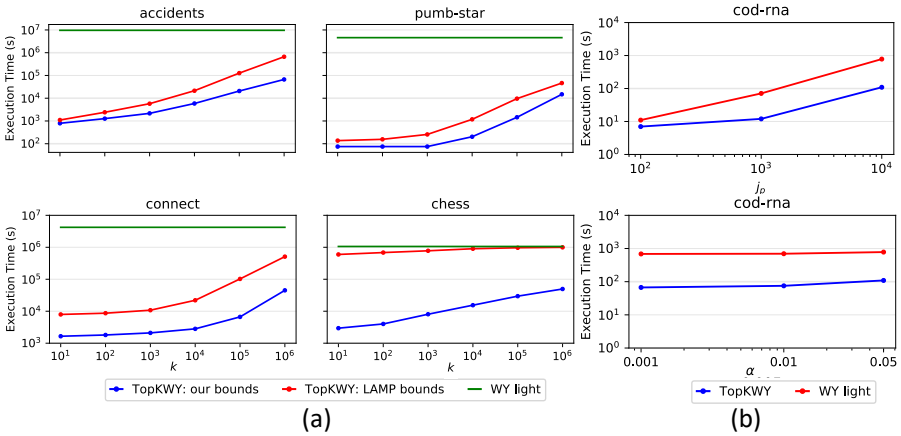

Fig. 6 (a) Comparison between the running time of WYlight and the running time of TopKWY using our improved bound $\psi'(\cdot)$ and the LAMP bound $\hat{\psi}(\cdot)$. (b) Running time for different values of α and j_p.

respect to WYlight, even using the same LAMP bounds. When our novel bound $\psi'(\cdot)$ is used in TopKWY we observe additional speed-ups, for a total up to more than two orders of magnitude. Therefore, the reduction in the number of patterns that need to be tested on the permuted datasets, obtained by the exploration strategy of TopKWY and our improved bound, is a crucial component for the performance of TopKWY.

Finally, we assessed the impact of α and j_p on the running time of TopKWY and WYlight on two representative datasets, cod-rna and accidents, which are representative for the two scenarios of a small number of significant patterns (cod-rna) and of a large number of significant patterns (accidents). In these experiments we fixed $k = 10^4$. Figure 6(b) reports the results for cod-rna. Results for accidents are not reported since the running time of accidents remained essentially the same for all values of α and j_p. This means that for accidents using the bounds introduced in Section 4 the computational effort is dominated by the pattern space exploration (and not the evaluation of the permuted datasets): considering only the top-k significant patterns is therefore crucial to analyze such dataset. For cod-rna, we observe that varying α has some but small impact on the runtime of both methods while there is a linear dependence of the running time of WYlight on j_p and a similar but less pronounced dependence of TopKWY. In all cases, TopKWY is faster than WYlight (for accidents WYlight does not terminate within 1 month) showing the efficiency of TopKWY for different ranges of the α and j_p parameters.

Comparison between Best First and Depth First strategies. We investigate the impact of the best first strategy adopted by TopKWY on the computational performances of the mining tasks. To do so, we compared TopKWY with TopKWY-dfs, the variant of TopKWY, which explores patterns in depth

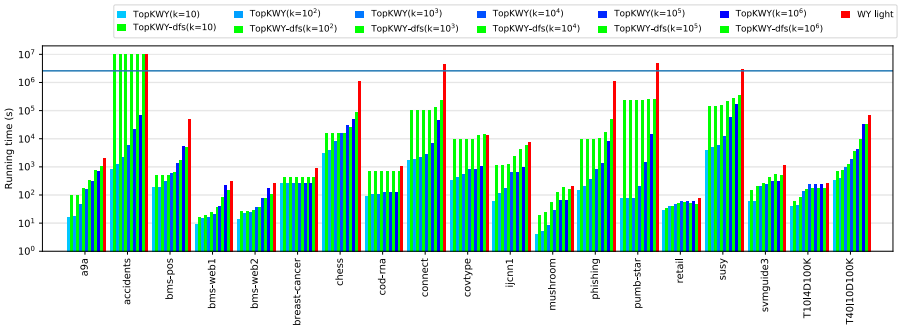

Fig. 7 Running time for TopKWY, TopKWY-dfs (with various values of k) and WYlight. The blue horizontal lines corresponds to 1 month of computation.

Table 3 a) Values of $g^\star$ (second row of the Table) computed by TopKWY* for different values of k (first row of the Table) on breast-cancer dataset with $j_p = 10^4$ and $\alpha = 0.05$. b) Number of results $|\mathcal{P}|$ (second row of the Table) found by TopKWY-ζ on breast-cancer dataset with $j_p = 10^4$, $\alpha = 0.05$, and $k = 10^4$, for different values of ζ (first row of the Table).

a)

k	10	10^2	10^3	10^4
$g^\star$	2	11	163	2396

b)

ζ	0.01	0.05	0.1	0.25
$\lvert\mathcal{P}\rvert$	0	24	624	10^4

first order (see Section 5.3). We ran TopKWY-dfs on the same set of experiments described in Section 7.3. Figure 7 shows the running times of TopKWY, TopKWY-dfs, and WYlight. We can clearly see that, for 9 datasets out of 19, there is a significant difference in the running times of TopKWY and TopKWY-dfs: this means that, in particular for smaller values of k, the exploration strategy is a critical component of TopKWY. For accidents, one of the most challenging dataset to analyze, both TopKWY-dfs and WYlight can not complete their execution in less than one month, even for $k = 10$. Therefore, for such dataset the best first strategy adopted by TopKWY is crucial.

Results for g-FWER We investigate the increase in statistical power of TopKWYwhen controlling the generalized-Family-Wise Error Rate (g-FWER) by analyzing datasets described in Section 7.3 having less than 10^6 results for $\alpha = 0.05$ when controlling the FWER. As we can see in Figure 8, for the breast-cancer dataset the number of significant patterns increased by more then two orders of magnitude as the value of g increases (i.e., when more false positives are allowed). Figure 9 show the running time of TopKWY when controlling the g-FWER at different values of g. As expected, the required time slightly increases but it stays practical for all datasets. We do not compare with other methods since TopKWY is the first algorithm to discover significant patterns with a rigorous control on the g-FWER.

In Table 3.a we show the computed values of $g^\star$ by TopKWY* (see Section 5.1 for its definition) on the breast-cancer dataset for $k \in$

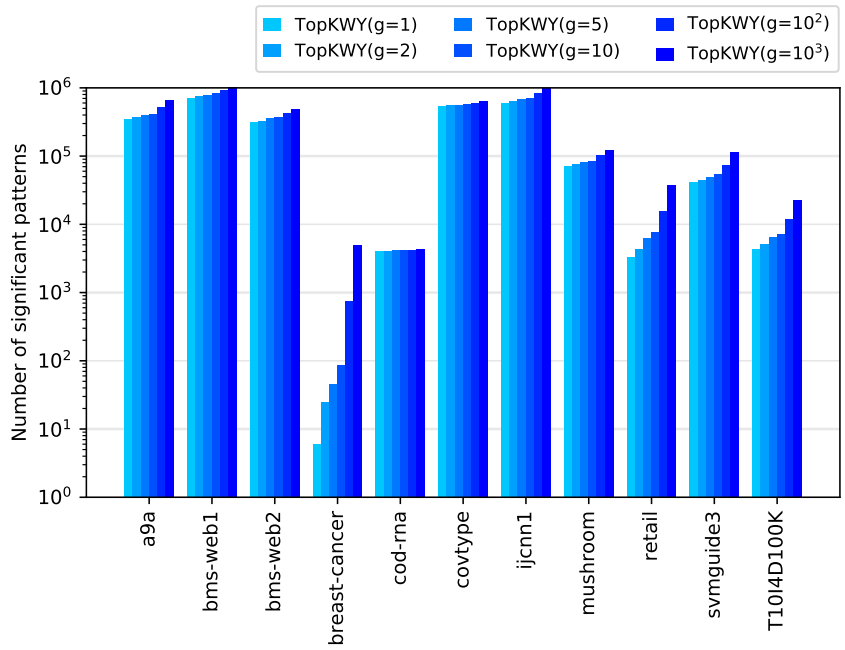

Fig. 8 Number of results found using TopKWY when controlling the g-FWER, for various values of g, $j_p = 10^4$, $k = 10^6$, and $\alpha = 0.05$.

$\{10, 10^2, 10^3, 10^4\}$. We can see that TopKWY* is able to provide informative estimates of the quality of the reported set of k most significant patterns, in terms of the minimum g such that the g-$FWER(p^k)$ is $\leqslant \alpha$, without the need of fixing g a-priori. In Table 3.b we show the number of results found using TopKWY-ζ for $k = 10^4$ on the breast-cancer dataset, varying $\zeta \in \{0.01, 0.05, 0.1, 0.25\}$. From these results we can see that TopKWY-ζ is a very flexible tool to discover significant patterns with bounds on both the output size and the maximum ratio of false discoveries, providing improved statistical power in situations where the number of significant patterns when controlling the $FWER$ is very low. (We do not show the running times for TopKWY* and TopKWY-ζ since those are very similar to the ones reported in Figure 9.)

7.4.2 Subgraphs Mining

We ran TopKWY-dfs on the datasets described in Section 7.2. Table 2 reports the number of significant patterns for $\alpha = 0.05$ in the datasets we considered, obtained by running TopKWY-dfs (with $k = +\infty$) or WYlight. For some datasets we stopped the computation after 1 month, so only a lower bound is available. In most cases, the number of significant patterns is extremely large: for 6 out of 11 datasets there are $> 5 \times 10^5$ significant patterns and in

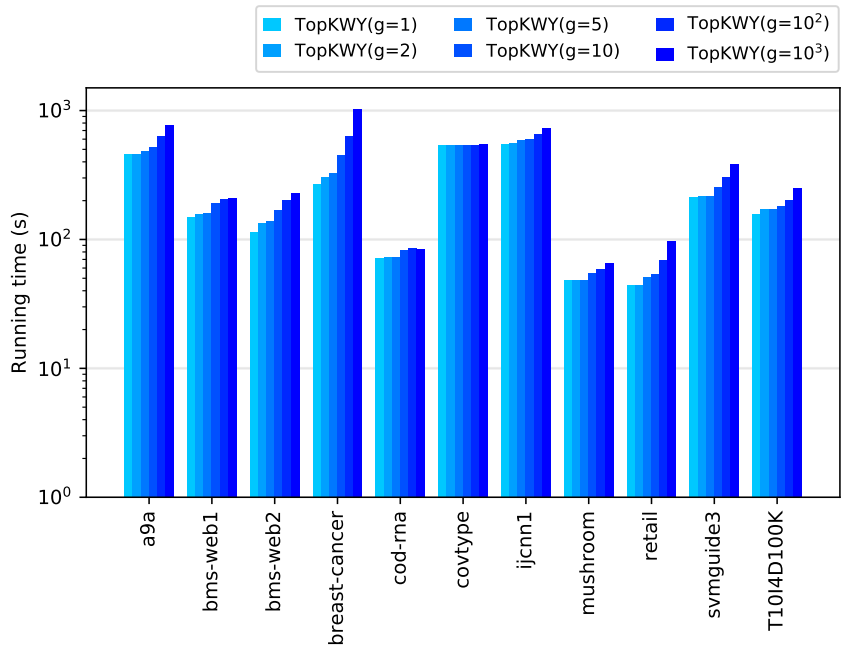

Fig. 9 Running time for TopKWY when controlling the g-FWER, for various values of g, $j_p = 10^4$, $k = 10^6$, and $\alpha = 0.05$.

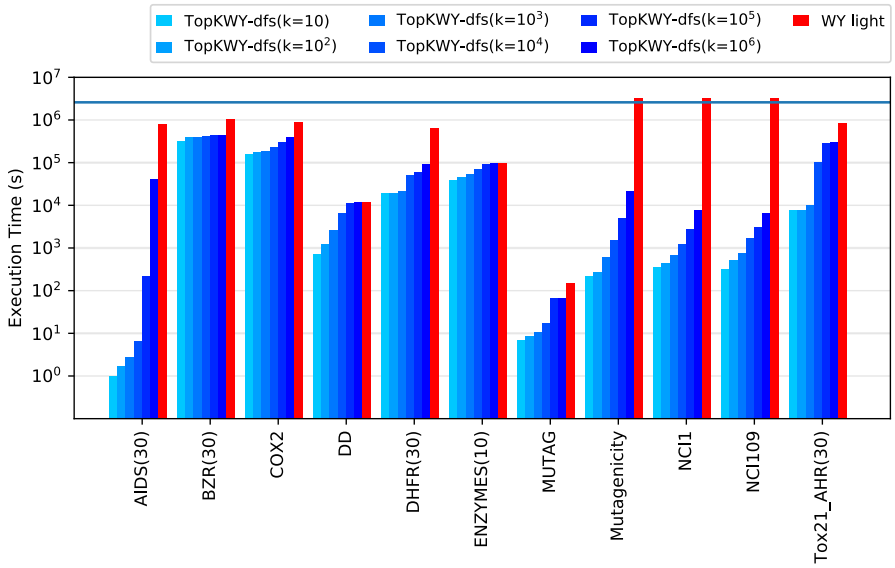

Fig. 10 Running time for TopKWY-dfs (with various values of k) and WYlight for significant subgraph mining. The blue horizontal line corresponds to 1 month of computation.

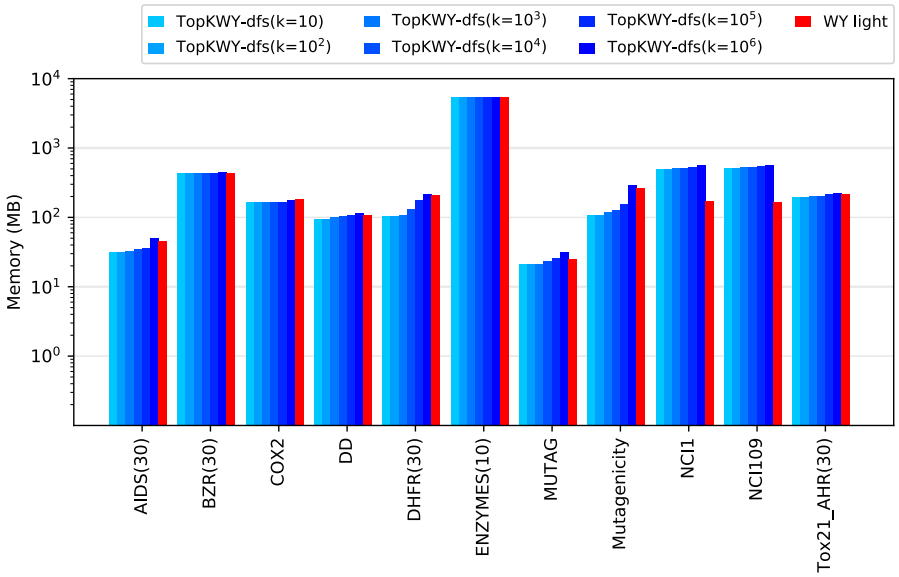

Fig. 11 Memory usage for TopKWY-dfs (with various values of k) and WYlight for significant subgraph mining.

5 datasets there are $> 10^6$ significant patterns. This shows that a direct way to limit the number of significant patterns in output is required for subgraphs mining as well.

We then compared the running time and memory requirement of TopKWY-dfs and of WYlight. Figure 10 compares the running times of TopKWY-dfs and WYlight. As for itemsets mining, when the number of significant patterns is lower than k, TopKWY-dfs finds all of them, obtaining the same output as WYlight. We can see that TopKWY-dfs is, in all cases, faster than WYlight: for 9 datasets out of 11 and for $k = 10^6$, TopKWY-dfs improves the running time by a factor at least 2. It is interesting to note that for 5 datasets the number of significant results is $< 10^6$, therefore TopKWY-dfs is faster even if its output is the same of WYlight. For 3 datasets the running time is reduced by more than two orders of magnitude, and WYlight is not able to terminate in less than 1 month. We can observe that these three datasets contains more than 10^6 significant results; this clearly shows that focusing on the most significant patterns leads to significant computational advantages and enables the analysis of such datasets.

Figure 11 compares the memory usage of TopKWY-dfs and WYlight. Both algorithms are very memory efficient and, while TopKWY-dfs usually requires more memory than WYlight, the difference is small: for 7 of the 8 datasets where WYlight terminates, TopKWY-dfs never requires more than 8% of the memory of WYlight, and 23% in the case of MUTAG, where the difference is of few MBs. (For the three datasets were WYlight does not terminate, we only report a lower bound to its memory usage.)

8 Conclusion

In this work we introduce TopKWY, an efficient algorithm to identify the top-k significant patterns with rigorous guarantees on the FWER and provide theoretical evidence of its effectiveness. Our extensive experimental evaluation shows that TopKWY enables the identification of significant patterns on large datasets and that it significantly improves over the state-of-the-art.

Our notion of top-k significant patterns and our algorithm TopKWY could be relevant to other mining problems, for example statistical emerging pattern mining (Komiyama et al., 2017), while providing a bound on the FWER. While we focus on bounding the FWER, a different approach would be to bound the false discovery rate (FDR) (Benjamini and Hochberg, 1995), that is the expected ratio of false discoveries among all reported patterns. Bounding the FDR is crucial in cases where the number of significant results obtained bounding the FWER is low. In addition, fully processing extremely large datasets may not be feasible: the combination of the techniques we develop in this work with sampling (e.g., the recently developed techniques by Pellegrina et al. (2019b) to extend TopKWY for the analysis of random samples) is a promising direction that we will investigate in future work.

Acknowledgements This work is supported, in part by the National Science Foundation grant IIS-1247581 (`https://www.nsf.gov/awardsearch/showAward?AWD_ID=1247581`), by the University of Padova grants SID2017 and STARS: Algorithms for Inferential Data Mining, and by MIUR, the Italian Ministry of Education, University and Research, under PRIN Project n. 20174LF3T8 AHeAD (Efficient Algorithms for HArnessing Networked Data).

References

Agrawal, R., Imieliński, T., and Swami, A. (1993). Mining association rules between sets of items in large databases. *SIGMOD Rec.*, 22:207–216.

Agrawal, R. and Srikant, R. (1994). Fast algorithms for mining association rules in large databases. In *Proc. 20th Int. Conf. Very Large Data Bases*, VLDB '94, pages 487–499, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Atzmueller, M. (2015). Subgroup discovery. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 5(1):35–49.

Bayardo Jr, R. J. (1998). Efficiently mining long patterns from databases. *ACM Sigmod Record*, 27(2):85–93.

Benjamini, Y. and 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)*, pages 289–300.

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.

Dong, G. and Bailey, J. (2012). *Contrast data mining: concepts, algorithms, and applications*. CRC Press.

Duivesteijn, W. and Knobbe, A. (2011). Exploiting false discoveries–statistical validation of patterns and quality measures in subgroup discovery. In *2011 IEEE 11th International Conference on Data Mining*, pages 151–160. IEEE.

Fisher, R. A. (1922). On the interpretation of χ 2 from contingency tables, and the calculation of p. *Journal of the Royal Statistical Society*, 85(1):87–94.

Gionis, A., Mannila, H., Mielikäinen, T., and Tsaparas, P. (2007). Assessing data mining results via swap randomization. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 1(3):14.

Hämäläinen, W. (2012). Kingfisher: an efficient algorithm for searching for both positive and negative dependency rules with statistical significance measures. *Knowledge and information systems*, 32(2):383–414.

Hämäläinen, W. and Webb, G. I. (2019). A tutorial on statistically sound pattern discovery. *Data Mining and Knowledge Discovery*, 33(2):325–377.

Han, J., Cheng, H., Xin, D., and Yan, X. (2007). Frequent pattern mining: current status and future directions. *Data Mining and Knowl. Disc.* , 15:55–86.

Han, J., Pei, J., and Yin, Y. (2000). Mining frequent patterns without candidate generation. In Chen, W., Naughton, J. F., and Bernstein, P. A., editors, *SIGMOD Conf.*, pages 1–12. ACM.

Han, J., Wang, J., Lu, Y., and Tzvetkov, P. (2002). Mining top-k frequent closed patterns without minimum support. In *Data Mining, 2002. ICDM 2003. Proceedings. 2002 IEEE International Conference on*, pages 211–218. IEEE.

Herrera, F., Carmona, C. J., González, P., and Del Jesus, M. J. (2011). An overview on subgroup discovery: foundations and applications. *Knowledge and information systems*, 29(3):495–525.

Komiyama, J., Ishihata, M., Arimura, H., Nishibayashi, T., and Minato, S.-i. (2017). Statistical emerging pattern mining with multiple testing correction. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 897–906. ACM.

Lehmann, E. L. and Romano, J. P. (2012). Generalizations of the familywise error rate. In *Selected Works of EL Lehmann*, pages 719–735. Springer.

Li, J., Liu, J., Toivonen, H., Satou, K., Sun, Y., and Sun, B. (2014). Discovering statistically non-redundant subgroups. *Knowledge-Based Systems*, 67:315–327.

Llinares-López, F., Sugiyama, M., Papaxanthos, L., and 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*, pages 725–734. ACM.

Minato, S.-i., Uno, T., Tsuda, K., Terada, A., and 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*, pages 422–436. Springer.

Nijssen, S. and Kok, J. N. (2004). A quickstart in frequent structure mining can make a difference. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 647–652. ACM.

Nijssen, S. and Kok, J. N. (2006). Frequent subgraph miners: runtimes don't say everything. *MLG 2006*, page 173.

Papaxanthos, L., Llinares-López, F., Bodenham, D., and Borgwardt, K. (2016). Finding significant combinations of features in the presence of categorical covariates. In *Advances in Neural Information Processing Systems*, pages 2279–2287.

Pasquier, N., Bastide, Y., Taouil, R., and Lakhal, L. (1999). Discovering frequent closed itemsets for association rules. In *International Conference on Database Theory*, pages 398–416. Springer.

Pellegrina, L., Riondato, M., and Vandin, F. (2019a). Hypothesis testing and statistically-sound pattern mining. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 3215–3216. ACM.

Pellegrina, L., Riondato, M., and Vandin, F. (2019b). Spumante: Significant pattern mining with unconditional testing. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining-KDD*, volume 19.

Pellegrina, L. and Vandin, F. (2018). Efficient mining of the most significant patterns with permutation testing. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2070–2079. ACM.

Pietracaprina, A. and Vandin, F. (2007). Efficient incremental mining of top-K frequent closed itemsets. In *Discovery Science*, volume 4755 of *Lecture Notes in Computer Science*, pages 275–280. Springer Berlin Heidelberg.

Tarone, R. (1990). A modified bonferroni method for discrete data. *Biometrics*, pages 515–522.

Terada, A., Kim, H., and 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*, pages 17–26. ACM.

Terada, A., Okada-Hatakeyama, M., Tsuda, K., and Sese, J. (2013a). Statistical significance of combinatorial regulations. *Proceedings of the National Academy of Sciences*, 110(32):12996–13001.

Terada, A., Tsuda, K., et al. (2016). Significant pattern mining with confounding variables. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 277–289. Springer.

Terada, A., Tsuda, K., and Sese, J. (2013b). Fast westfall-young permutation procedure for combinatorial regulation discovery. In *Bioinformatics and Biomedicine (BIBM), 2013 IEEE International Conference on*, pages 153–158. IEEE.

Uno, T., Kiyomi, M., and Arimura, H. (2005). Lcm ver. 3: collaboration of array, bitmap and prefix tree for frequent itemset mining. In *Proceedings of the 1st international workshop on open source data mining: frequent pattern*

mining implementations, pages 77–86. ACM.

van der Laan, M. J., Dudoit, S., and Pollard, K. S. (2004). Augmentation procedures for control of the generalized family-wise error rate and tail probabilities for the proportion of false positives. *Statistical applications in genetics and molecular biology*, 3(1):1–25.

Webb, G. I. (2006). Discovering significant rules. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 434–443. ACM.

Webb, G. I. (2007). Discovering significant patterns. *Machine learning*, 68(1):1–33.

Webb, G. I. (2008). Layered critical values: a powerful direct-adjustment approach to discovering significant patterns. *Machine Learning*, 71(2-3):307–323.

Westfall, P. H. and Young, S. S. (1993). Resampling-based multiple testing: Examples and methods for p-value adjustment. *Wiley Series in Probability and Statistics*.

Wörlein, M., Meinl, T., Fischer, I., and Philippsen, M. (2005). A quantitative comparison of the subgraph miners mofa, gspan, ffsm, and gaston. In *European Conference on Principles of Data Mining and Knowledge Discovery*, pages 392–403. Springer.

Zandolin, D. and Pietracaprina, A. (2003). Mining frequent itemsets using patricia tries. In *Proceedings of FIMI03*, volume 90.