Efficient Mining of the Most Significant Patterns with Permutation Testing

Leonardo Pellegrina
Department of Information Engineering
Università di Padova
Padova, Italy
pellegri@dei.unipd.it

Fabio Vandin
Department of Information Engineering
Università di Padova
Padova, Italy
fabio.vandin@unipd.it

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 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.

CCS CONCEPTS

• **Mathematics of computing → Probabilistic inference problems**; • **Information systems → Data mining**;

KEYWORDS

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

ACM Reference Format:
Leonardo Pellegrina and Fabio Vandin. 2018. Efficient Mining of the Most Significant Patterns with Permutation Testing. In *KDD '18: The 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, August 19–23, 2018, London, United Kingdom.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3219819.3219997

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 [9]. In its original definition [1] it requires to identify patterns that appear in a fraction at least σ of all the transactions of a transactional dataset.

ACM ISBN 978-1-4503-5552-0/18/08…$15.00
https://doi.org/10.1145/3219819.3219997

Significant pattern mining [6] 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 [7]), 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 m patterns are tested we expect αm 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* [5], is to adjust the significance threshold by dividing α by the number m 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 [26–28]. 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. [22], that proposes LAMP, the first method to identify significant patterns without limiting their size. LAMP is based on the work by Tarone [20], 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. [15] 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 [29] have been proposed, first by Terada et al. [24] with FastWY and then by Llinares-López et al. [14] 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 LAMP 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., in [14, 24]) to partially alleviate this problem is to consider only closed patterns [9], 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. The use of the Westfall-Young permutation test allows TopKWY to have higher power when the number of significant patterns is small, while reporting only the most significant patterns when the number of such patterns is huge. TopKWY is based on a exploration strategy similar to the one used by TopKMiner [19], 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 conduct an extensive experimental evaluation of the use of TopKWY to extract significant itemsets, 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 [1], a number of methods have been developed to efficiently extract all frequent patterns (see [9] 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 [18] or maximal patterns [3], have been designed. Methods that directly limit the number of patterns by reporting the k most frequent closed patterns have been designed [11, 19] as well.

Many methods have been developed for *subgroups discovery* (see, e.g., the reviews by Herrera et al. [12] and by Atzmueller [2]), 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, but do not assess the statistical significance of the association or do not correct for multiple hypothesis testing.

In addition to the contributions for significant pattern mining mentioned above [14, 15, 21, 22, 24] (described more in depth in Section 2), recent work has extended the extraction of statistically sound patterns [8] in directions that are orthogonal to our contributions. [13] has developed an efficient technique for multiple testing correction in the mining of statistical emerging patterns. [17, 23] 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 defined on a universe $\mathcal{I}$ of f features. Each transaction 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. $n_1 \leq n/2$. We define a *pattern* S as a set of features $S = \{\ell_1, \ell_2, ..., \ell_k\}$, 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. Given a pattern S, we define its support x_S as the number of transactions containing S,

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

Figure 1: 2×2 **contingency table.**

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.

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 [7] 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:

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

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 $\mathcal{D}$:

$$p_S = p_S(a_S) = \sum_{k:\mathbf{Pr}(k|x_S,n_1,n) \leq Pr(a_S|x_S,n_1,n)} \mathbf{Pr}(k|x_S,n_1,n) \ . \quad (1)$$

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= $\mathbf{Pr}(FP > 0)$. 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 $\leq \delta$.

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 $FWER(\delta) \leq d\delta = \alpha$. 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 be 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:

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

In this regard, a key result from Tarone [20] is the introduction of *minimum attainable p-value*: if we use a corrected significance threshold δ, patterns whose p-value cannot be $\leq \delta$, called *untestable*, do not need to be explored and considered in the multiple-hypothesis correction. Therefore, if we define $k(\delta)$ as the number of patterns having minimum attainable p-value $\leq \delta$, then the adjusted Bonferroni correction when threshold δ is used can be written as $\delta = \alpha/k(\delta)$.

LAMP [22] introduced such concepts into the mining of significant patterns: 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 exist 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 1:

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

Given a pattern S, its support x_S, and a corrected significance threshold δ, if $\psi(x_S) > \delta$ the pattern S will never be significant and is therefore *untestable*. To use $\psi(x_S)$ in significant pattern mining, Terada et al. [22] introduced in LAMP a monotonically decreasing lower bound $\hat{\psi}(x_S)$ on $\psi(x_S)$:

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

Terada et al. 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

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

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 [29] 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 null hypotheses *joint distribution* 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:

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

where $\mathbb{1}(\cdot)$ 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 $\delta^* = \max_\delta\{FWER(\delta) \leq \alpha\}$ with $FWER(\delta)$ estimated as above.

The WY procedure 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. 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. [24] 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 it 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. [21], HWY, exploits a more efficient mining strategy and parallel computing to accelerate FastWY.

Llinares-López et al. [14] 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 $\{p_{\min}^{(j)}\}_{j=1}^{j_p}$ 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 [10] 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. At the same time, it 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 $\leq \sigma$, 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 $\leq \delta^*$. As shown in [14], 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 $\delta(\alpha) = \max_\delta\{FWER(\delta) \leq \alpha\}$ the threshold obtained through the WY permutation procedure for FWER bound 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 $\leq \alpha$, defined as:

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

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 2.1. *The set $TSP(\mathcal{D}, k, \alpha)$ has FWER $\leq \alpha$.*

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 3.3). Finally, we introduce some crucial implementation details (Section 3.4), focusing on the problem of mining significant itemsets.

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 [14] 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 [19] 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

[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 [9, 16, 25] and is required by WYlight as well.

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 3.3). 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 $\leq \delta_m$ 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 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 $S = \emptyset$: 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 $\geq \sigma$ in the queue Q. The details of an efficient implementation of `expand` are described in Section 3.4. The `while` loop (lines 6-22) 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 $< \delta_m$, then S is inserted in P in line 10. In line 8, σ' is set to x_S, which is the exact upper bound of 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 needs to compute the p-value of pattern S in the j_p permuted datasets, updating the values of $\{p^{(j)}_{\min}\}_{j=1}^{j_p}$ 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 to test it on the permuted datasets or not (see Section 3.3). This allows to skip the expensive computation of the j_p supports $\{a^{(j)}_S\}_{j=1}^{j_p}$ of S on the minority class of the permuted datasets for several patterns S.

The significance threshold δ_m and the corresponding minimum support threshold σ are increased during the exploration in two cases: when the estimated $FWER(\delta_m)$ for the current threshold δ_m increases above α (lines 15-16), or when more than k patterns with p-value $\leq \delta_m$ are observed (lines 17-19). 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 20, and elements which are not significant are removed from P in line 21. The current pattern S is expanded in line 22, and all its children having support $\geq \sigma$ are inserted into Q. The exploration ends when Q gets empty, or when the maximum support of all its elements is $< \sigma$. When this happens, all elements still contained in P with p-value at most δ_m are reported as significant.

We note that 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 considering a maximum value k^*, provided by the user, in Algorithm 1 to definitely prune untestable patterns, but freezing the computation after k patterns with p-value below the current value of σ 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 $D$ with class labels $c$, number of
       permutations $j_p$, target FWER $\alpha$, number of results $k$
Output: Set of top-$k$ significant patterns with FWER $\leq \alpha$
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($\emptyset$ , $Q$);
6  while ($Q \neq \emptyset$) and ($\max(x_S \in Q) \geq \sigma$) do
7      $(S, x_S, a_S, f_S) \leftarrow Q$.removeMax();
8      $\sigma' \leftarrow x_S$;
9      if $p_S \leq \hat{\psi}(\sigma)$ then
10         $P$.insert($S, p_S$);
       /* $O$ = patterns surely in $TSP(\mathcal{D}, k, \alpha)$ */
11     $O \leftarrow \{S' \in P : p_{S'} < \hat{\psi}(\sigma')\}|$; produce $O$ in output;
12     remove patterns in $O$ from $P$; $n_o \leftarrow |O|$; $k \leftarrow k - n_o$;
13     if $\psi'(x_S, f_S) \leq \hat{\psi}(\sigma)$ then
14         test($S$ , $\{p^{(j)}_{\min}\}_{j=1}^{j_p}$);
       /* update $\delta_m$ based on estimate of $\delta^*$ */
15     $\delta_m \leftarrow \min\{\delta_m, \max\{\delta : FWER(\delta) \leq \alpha\}\}$;
16     $\sigma \leftarrow \psi^{-1}(\delta_m)$;
       /* update $\delta_m$ based on top-$k$ patterns in $P$ */
17     $p^{(k)} \leftarrow k$-th largest $p$-value in $P$;
18     $\delta_m \leftarrow \min\{\delta_m, p^{(k)}\}$;
19     $\sigma \leftarrow \psi^{-1}(\delta_m)$;
       /* remove untestable patterns from $Q$ */
20     remove from $Q$ all patterns $S'$ with $x_{S'} < \sigma$;
       /* remove non-significant patterns from $P$ */
21     remove from $P$ all patterns $S'$ with $p_{S'} > \delta_m$;
22     expand($S$ , $Q$);
23 produce in output $\{S' \in P : p_{S'} \leq \delta_m\}$;
```

3.2 Analysis

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

Theorem 3.1. *[Correctness of TopKWY] TopKWY outputs the set $TSP(\mathcal{D}, k, \alpha)$ of top-k significant patterns with FWER $\leq \alpha$.*

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 $\geq \delta^*$. δ_m is decreased (and the corresponding minimum support threshold σ is increased) during the exploration in two cases. The first case (lines 15-16) 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 $FWER(\delta) \leq \alpha$. The second case is when more than k patterns with p-value $\leq \delta_m$ are observed (lines 17-19). 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 23. 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 $\geq \hat{\psi}(\sigma')$. 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 3.2 (Optimality of TopKWY). *TopKWY expands only patterns of support* $\geq \hat{\psi}^{-1}(\delta_m)$.

Proof. Similarly to the proof of Thm. 3.1, 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 $\geq \hat{\psi}(\sigma')$, that is $\sigma' \geq \hat{\psi}^{-1}(\delta_m)$ (i.e., S is testable). □

3.3 Improved Bound on Minimum Attainable p-value

TopKWY employs novel and efficiently computable lower bounds on the minimum p-value achievable by a pattern S. Let the pattern S be a super pattern of Y, that is $S \supset Y$. Then $x_Y \geq a_Y \geq 0$ and $x_Y \geq x_S$. 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:

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

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). Let

$$a_{Y_{\min}} = \min\left\{\{a_Y^{(j)}\}_{j=1}^{j_p}\right\}$$

and

$$a_{Y_{\max}} = \max\left\{\{a_Y^{(j)}\}_{j=1}^{j_p}\right\} \quad .$$

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

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

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 $\psi(x_S)$:

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

The following is a simple consequence of 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 3.3. $\psi'(x_S, x_Y, a_{Y_{\min}}, a_{Y_{\max}}) \geq \psi(x_S)$.

If for the current 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 $\{p_{\min}^{(j)}\}$ and therefore cannot contribute to the computation of $\delta(\alpha)$. 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 an high degree of correlation.

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 $\leq x_S - 1$, 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 optimization allows to 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.

3.4 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. We now present some key implementation details for the case of significant itemsets mining.

Dataset representation. TopKWY uses a PatriciaTrie [30] 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 Patricia Trie. 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. A technique similar to reservoir sampling is used to generate the j_p permuted labels of every transaction. (All details will be provided in the extended version of the paper.)

P-values representation and look-up. The precise ranking of observed p-values is a fundamental prerequisite when mining the top-k most significant patterns. Significant pattern mining algorithms (e.g., [14]) typically suffer from underflow imprecision of very low p-values, inevitably due to the finite representation of float values. When this happens, patterns are reported as significant with a p-value equal to 0. This imprecision does not allow the ranking of those p-values, nor the restriction to the top-k most significant patterns. Therefore, TopKWY relies on log-p-values: this representation do not suffer from the same underflow problem. A drawback of this design choice is a slightly increase in computation time of p-values, since the sum of the hypergeometric tails involves the sum of log-probabilities. To reduce this issue, TopKWY (similarly to [21]) stores computed p-values in a lookup table of fixed size, so that the time required to compute p-values is significantly reduced with a minimum memory overhead.

4 EXPERIMENTAL EVALUATION

We implemented and tested TopKWY for the extraction of significant itemsets. 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 with the state-of-the-art tool for significant pattern mining, WYlight [14]. We do not compare with LAMP [22] or derived strategies [15] since [14] 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 4.1 we describe the implementation and computational environment for our experiments. In Section 4.2 we describe the datasets we used. In Section 4.3 we describe the experiments we have performed and our choice of parameters. Finally, in Section 4.4 we report and discuss the results of our experiments.

4.1 Implementation and Environment

We implemented TopKWY in C/C++. The TopKMiner algorithm [19] is used to process closed itemsets in TopKWY. Our code is available at https://github.com/VandinLab/TopKWY. For WYlight we used the C/C++ implementation (based on LCM [25]) made available by the authors at https://github.com/fllinares/wylight. Both implementations were compiled with the C++ `gcc 4.8.4` compiler. Our experiments have been performed on a 16-core 2.30 GHz Intel Xeon CPU machine with 512 GB of RAM, running on Ubuntu 14.04. Scripts to replicate all experiments described in the paper are available at https://github.com/VandinLab/TopKWY.

4.2 Datasets

We performed our experiments using 19 datasets: the 10 largest ones used in [14] and available at FIMI'04[2] and UCI[3], all the datasets used in [13], available from the libSVM repository[4], and 4 additional ones

[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/

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	$> 10^7$

Table 1: 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 $FWER = 0.05$.

(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 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 [13]).

4.3 Parameters and Experiments

For TopKWY we considered $k = 10, 10^2, 10^3, 10^4, 10^5, 10^6$. For all the datasets we analyzed, we ran TopKWY, 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 [14], and fixed the commonly used value $\alpha = 0.05$ as FWER threshold. For the comparison between TopKWY and WYlight we repeated every experiment 10 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 WYlight, we only report

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

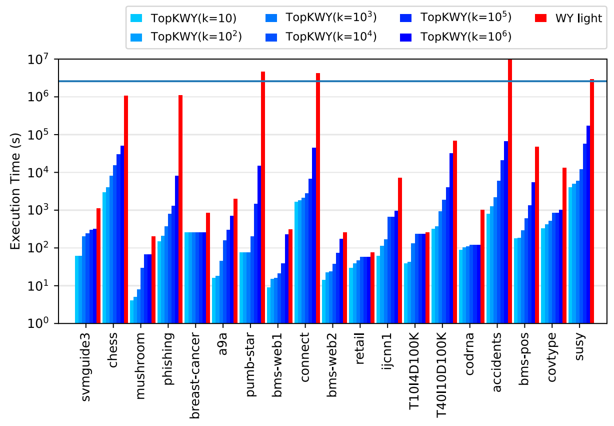

Figure 2: Running time for TopKWY (with various values of k) and WYlight. The blue horizontal lines corresponds to 1 month of computation.

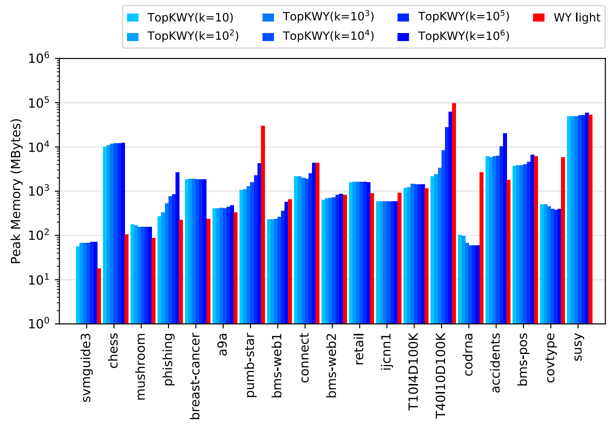

Figure 3: Peak memory for TopKWY (with various values of k) and WYlight.

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 the runtime required by WYlight. 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.

4.4 Results

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 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 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 ($\leq 20GBytes$) 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*). 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 3.4) 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

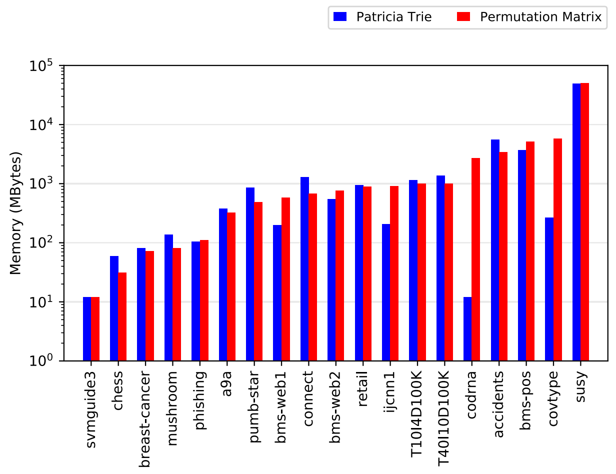

Figure 4: Memory requirement for permuted class labels using PatriciaTrie and permutation matrix.

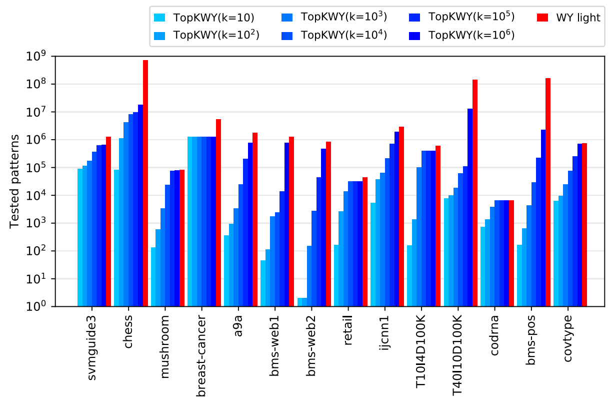

Figure 5: Comparison between the number of tested patterns on the permuted datasets by TopKWY and WYlight.

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 3.3) 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 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.

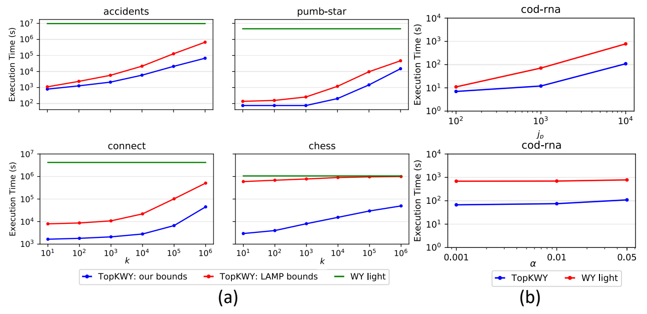

Figure 6: (a) Comparison between the running time of WYlight and the running time of TopKWY using our improved bound $\psi'(\cdot)$ and the LAMP bound $\hat{\psi}(\cdot)$. (b) Running time for different values of α and j_p.

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 3.3 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.

5 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 [13], 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) [4], that is the expected ratio of false discoveries among all reported patterns. This is an interesting direction for future research in which the top-k approach we propose is crucial, since more patterns can be reported with FDR $\leq \alpha$ than with FWER $\leq \alpha$ and our experiments show that in many cases a large number of patterns is reported with FWER $\leq \alpha$. In addition, fully processing extremely large datasets may not be feasible: the combination of the techniques we develop in this work with sampling is a promising direction that we will investigate in future work.

ACKNOWLEDGMENTS

This work is supported, in part by the National Science Foundation grant IIS-1247581 (https://www.nsf.gov/awardsearch/showAward?AWD_ID=1247581) and by the University of Padova grants SID2017 and STARS: Algorithms for Inferential Data Mining.

REFERENCES

[1] Rakesh Agrawal, Tomasz Imieliński, and Arun Swami. 1993. Mining association rules between sets of items in large databases. *SIGMOD Rec.* 22 (June 1993), 207–216. Issue 2. https://doi.org/10.1145/170036.170072

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

[3] Roberto J Bayardo Jr. 1998. Efficiently mining long patterns from databases. *ACM Sigmod Record* 27, 2 (1998), 85–93.

[4] Yoav Benjamini and Yosef Hochberg. 1995. Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the royal statistical society. Series B (Methodological)* (1995), 289–300.

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

[6] Guozhu Dong and James Bailey. 2012. *Contrast data mining: concepts, algorithms, and applications.* CRC Press.

[7] Ronald A Fisher. 1922. On the interpretation of χ 2 from contingency tables, and the calculation of P. *Journal of the Royal Statistical Society* 85, 1 (1922), 87–94.

[8] Wilhelmiina Hämäläinen and Geoff Webb. 2017. A Tutorial on Statistically Sound Pattern Discovery. (09 2017). arXiv:1709.03904 https://arxiv.org/abs/1709.03904

[9] Jiawei Han, Hong Cheng, Dong Xin, and Xifeng Yan. 2007. Frequent pattern mining: current status and future directions. *Data Mining and Knowl. Disc.* 15 (2007), 55–86. Issue 1. https://doi.org/10.1007/s10618-006-0059-1

[10] Jiawei Han, Jian Pei, and Yiwen Yin. 2000. Mining Frequent Patterns without Candidate Generation. In *SIGMOD Conf.*, Weidong Chen, Jeffrey F. Naughton, and Philip A. Bernstein (Eds.). ACM, 1–12.

[11] Jiawei Han, Jianyong Wang, Ying Lu, and Petre Tzvetkov. 2002. Mining top-k frequent closed patterns without minimum support. In *Data Mining, 2002. ICDM 2003. Proceedings. 2002 IEEE International Conference on.* IEEE, 211–218.

[12] Franciso Herrera, Cristóbal José Carmona, Pedro González, and María José Del Jesus. 2011. An overview on subgroup discovery: foundations and applications. *Knowledge and information systems* 29, 3 (2011), 495–525.

[13] Junpei Komiyama, Masakazu Ishihata, Hiroki Arimura, Takashi Nishibayashi, and Shin-ichi Minato. 2017. Statistical Emerging Pattern Mining with Multiple Testing Correction. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.* ACM, 897–906.

[14] Felipe Llinares-López, Mahito Sugiyama, Laetitia Papaxanthos, and Karsten Borgwardt. 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.* ACM, 725–734.

[15] Shin-ichi Minato, Takeaki Uno, Koji Tsuda, Aika Terada, and Jun Sese. 2014. A fast method of statistical assessment for combinatorial hypotheses based on frequent itemset enumeration. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases.* Springer, 422–436.

[16] Siegfried Nijssen and Joost N Kok. 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.* ACM, 647–652.

[17] Laetitia Papaxanthos, Felipe Llinares-López, Dean Bodenham, and Karsten Borgwardt. 2016. Finding significant combinations of features in the presence of categorical covariates. In *Advances in Neural Information Processing Systems.* 2279–2287.

[18] Nicolas Pasquier, Yves Bastide, Rafik Taouil, and Lotfi Lakhal. 1999. Discovering frequent closed itemsets for association rules. In *International Conference on Database Theory.* Springer, 398–416.

[19] Andrea Pietracaprina and Fabio Vandin. 2007. Efficient Incremental Mining of Top-K Frequent Closed Itemsets. In *Discovery Science.* Lecture Notes in Computer Science, Vol. 4755. 275–280.

[20] RE Tarone. 1990. A modified Bonferroni method for discrete data. *Biometrics* (1990), 515–522.

[21] Aika Terada, Hanyoung Kim, and Jun Sese. 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.* ACM, 17–26.

[22] Aika Terada, Mariko Okada-Hatakeyama, Koji Tsuda, and Jun Sese. 2013. Statistical significance of combinatorial regulations. *Proceedings of the National Academy of Sciences* 110, 32 (2013), 12996–13001.

[23] Aika Terada, Koji Tsuda, et al. 2016. Significant pattern mining with confounding variables. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining.* Springer, 277–289.

[24] Aika Terada, Koji Tsuda, and Jun Sese. 2013. Fast Westfall-Young permutation procedure for combinatorial regulation discovery. In *Bioinformatics and Biomedicine (BIBM), 2013 IEEE International Conference on.* IEEE, 153–158.

[25] Takeaki Uno, Tatsuya Asai, Yuzo Uchida, and Hiroki Arimura. 2004. An efficient algorithm for enumerating closed patterns in transaction databases. In *International Conference on Discovery Science.* Springer, 16–31.

[26] Geoffrey I Webb. 2006. Discovering significant rules. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining.* ACM, 434–443.

[27] Geoffrey I Webb. 2007. Discovering significant patterns. *Machine learning* 68, 1 (2007), 1–33.

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

[29] Peter H Westfall and Stanley S Young. 1993. Resampling-Based Multiple Testing: Examples and Methods for p-Value Adjustment (Wiley Series in Probability and Statistics). (1993).

[30] D Zandolin and A Pietracaprina. 2003. Mining frequent itemsets using patricia tries. In *Proceedings of FIMI03*, Vol. 90.