Weighted Statistically Significant Pattern Mining

Tingfu Zhou
Jinan University
Guangzhou, China
tfzhou@gmail.com

Zhenlian Qi
Guangdong Eco-Engineering Polytechnic
Guangzhou, China
qzlhit@gmail.com

Wensheng Gan*
Jinan University
Guangzhou, China
wsgan001@gmail.com

Shicheng Wan
Guangdong University of Technology
Guangzhou, China
scwan1998@gmail.com

Guoting Chen
Harbin Institute of Technology
Shenzhen, China
chenguoting@hit.edu.cn

ABSTRACT

Pattern discovery (aka pattern mining) is a fundamental task in the field of data science. Statistically significant pattern mining (SSPM) is the task of finding useful patterns that statistically occur more often from databases for one class than for another. The existing SSPM task does not consider the weight of each item. While in the real world, the significant level of different items/objects is various. Therefore, in this paper, we introduce the **W**eighted **S**tatistically **S**ignificant **P**atterns **M**ining (WSSPM) problem and propose a novel **WSSpm** algorithm to successfully solve it. We present a new framework that effectively mines weighted statistically significant patterns by combining the weighted upper-bound model and the multiple hypotheses test. We also propose a new weighted support threshold that can satisfy the demand of WSSPM and prove its correctness and completeness. Besides, our weighted support threshold and modified weighted upper-bound can effectively shrink the mining range. Finally, experimental results on several real datasets show that the WSSpm algorithm performs well in terms of execution time and memory storage.

KEYWORDS

pattern mining, multiple hypothesis testing, significant pattern, weighted pattern.

ACM Reference Format:
Tingfu Zhou, Zhenlian Qi, Wensheng Gan, Shicheng Wan, and Guoting Chen. 2023. Weighted Statistically Significant Pattern Mining. In *Companion Proceedings of the ACM Web Conference 2023 (WWW '23 Companion), April 30-May 4, 2023, Austin, TX, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3543873.3587586

*Corresponding author, also with Pazhou Lab, Guangzhou 510330, China

WWW '23 Companion, April 30-May 4, 2023, Austin, TX, USA

ACM ISBN 978-1-4503-9419-2/23/04…$15.00
https://doi.org/10.1145/3543873.3587586

1 INTRODUCTION

Rich data is now generated from various platforms, including labs, network data, and e-commerce platforms. As a result, data mining [12, 42] has attached an important role in discovering potential patterns from massive data. During the past few decades, pattern mining [13] has been one of the most attractive fields in data mining. For example, the frequent pattern mining task is identifying the patterns that occur frequently from transaction datasets [21]. The significant patterns have a significant association with binary class [7]. Different from traditional frequent pattern mining frameworks which use the associate rule [1, 18] to discover objective patterns, statistically significant pattern mining (SSPM) evaluates the p-value of the hypothesis testing to discover objective patterns. With its statistical accuracy, SSPM has been widely applied in various fields [20, 26]. For instance, a manager of a beverage company wants to investigate the potential customers of a tea product, so he needs to accurately advertise target customers (such as students and workers). In this case, the manager can use SSPM to make the decision: the students are the potential customer under the null hypothesis and the workers are the potential customer under the alternative hypothesis. SSPM can calculate the p-value of the assumption that the null hypothesis is true. Using the p-value, the manager can find the potential customer with statistical accuracy.

In SSPM, a critical issue is the multiple hypothesis problem. That is, the probability of finding $\alpha \times n$ patterns whose p-values are lower than the pre-defined threshold α while testing n patterns. Recent work has provided multiple methods for solving this problem, e.g., Bonferroni correction, Tarone's trick [38], and Westfall-Young permutation method [44]. The Westfall-Young permutation method solves the multiple hypothesis problem by randomly labeling the class of the transactions. Then, Llinares-López *et al.* [29] proposed a heuristic algorithm named WYlight, which utilizes the Westfall-Young permutation method. Up to now, WYlight is still the state-of-the-art SSPM algorithm. However, different items have different significance levels. For example, in a beverage company, the profit of different products is varied. The profit of a bottle of wine is higher than the profit of a bottle of mineral water. Existing SSPM algorithms do not consider this weighted situation. Besides, existing weighted support thresholds, like the absolute weighted support of WSpan [46] or the relative weighted support of IUA [25] can not meet the needs of SSPM since the minimum frequent threshold will change with the multiple hypothesis error control

in SSPM. The upper-bound model of weighted pattern mining has been studied by many researchers. Classic weighted upper bound like maximum item bound [45] or IUA [25] has been widely used [22]. The state-of-the-art weighted upper-bound model is WFSPM [23]. WFSPM sets use the highest possible weighted support as the weighted upper bound. Compared to other weighted upper-bound models [25, 46], WFSPM is tighter and more efficient. In SSPM, the minimum frequency will change with the control process of the multiple-hypothesis problem. Thus, these weighted upper bounds can not match SSPM.

Therefore, in light of this lack of weight problem in statistically significant pattern mining, we propose a weighted statistically significant pattern mining algorithm (WSSpm for short). We modify the outline to calculate the outline of the Westfall-Young permutation of WYlight [29] so its calculation process will be more convenient and faster. In addition, in order to add a weighted pattern mining function, we propose a new weighted support threshold that can satisfy the statistical needs of SSPM and accommodate the existing weighted upper-bound model so that it can fit the outline of SSPM. The main contributions to this work are as follows:

- We are the first to formulate the problem of weighted statistically significant pattern mining (WSSPM). We also propose the WSSpm algorithm to excavate the objective patterns accurately and efficiently. A new weighted support threshold can satisfy the need for SSPM.
- Our WSSpm algorithm combines the weighted support threshold, weighted upper-bound model, and the multiple hypotheses test to accommodate WSSPM. The correctness of WSSpm is given in detail.
- We conduct several experiments on real datasets to examine the effectiveness and efficiency of WSSpm. It shows that WSSpm has the additional weighted value function without losing efficiency in execution time or memory storage. Besides, the weighted support threshold and upper bound can effectively shrink the mining range.

The remainder of this paper is organized as follows: Section 2 introduces the related work in short. Section 3 states the related definitions and the description of the addressed problem. Section 4 introduces the proposed WSSpm algorithm. The experimental results and analysis are presented in Section 5. Finally, the summary is discussed in Section 6.

2 RELATED WORK

2.1 Statistical significant pattern mining

Pattern mining [10, 13] is a fundamental task in knowledge discovery from data, including frequent pattern mining [14], high-utility pattern mining [16, 47], weighted pattern discovery [15], uncertain pattern mining [4], graph mining, and others. The study [19] is the first to consider significant pattern mining as a multiple-hypothesis problem and solves it by using Fisher's exact test. However, earlier work in SSPM did not consider the problem of correcting in multiple hypotheses tests [27]. To solve the problem of multiple hypotheses in SSPM, a standard method called the Bonferroni method [3] was proposed. Bonferroni adjusts the significant threshold according to dividing it by the number of tested patterns. Due to the huge number of patterns, this method has limited statistical power. In other words, it means only a few patterns can meet the condition of such a small threshold. Other methods like the algorithm of Duivesteijn and Knobbe [8] use a swap randomization technique of Gionis *et al.* [17] to deal with multiple hypothesis tests, or Webb [43] uses layered critical values to correct the multiple hypothesis testing. Terada *et al.* [39] used the work of Tarone [38] which shows that the untestable pattern, that is, the pattern with no probability to reach statistical significance, does not need to be calculated when correcting multiple hypotheses tests. Later, Minato *et al.* [30] expanded this method to identify testable patterns. However, this method still suffers from limited power due to its numerous testable patterns. Several algorithms based on the more efficient Westfall-Young permutation [44] were proposed. Terada *et al.* [40] proposed FastWY. Later, Linares-Lopez *et al.* [29] proposed the WYlight algorithm, which is the state-of-the-art algorithm in SSPM.

The test method is another important part of SSPM. Fisher's exact test, also known as the conditional exact test, is the mainstream of SSPM [19]. In Fisher's exact test, the total number of rows and columns of contingency is fixed. Thus, Fisher's exact test can simplify the calculation by removing redundant parameters. Instead of fixing the redundant parameter on the observation value as a condition, Barnard's test [31] puts the redundant parameter in the range of parameter space. Besides Fisher's exact test and Barnard's test, robust permutation test can also mine significant patterns based on a small sample [6]. However, Fisher's exact test is widely used in SSPM [20] because of its convenience in the calculation. Recent work has extended SSPM into various fields. Webb and Hämäläinen proposed a method that can discover statistically sound patterns [20]. Le *et al.* [26] used the outline of WYlight [29] to discover the sub-trajectory. WYlight was then modified to discover the top-k significant patterns [32]. There are also many significant pattern mining studies for discovering statistical emerging patterns [24] and high-utility patterns [36, 41].

2.2 Upper-bound in weighted pattern mining

In the real world, the significance (e.g., weight, risk, and utility) of different items/objects is various [16]. To explore rich data, the problem of weighted pattern mining was proposed [34]. Tao *et al.* [37] introduced the concepts of the mean value of an item's weight in a transaction and the weighted support, which is defined as the quotient of the sum of a transaction's weight values that contains the pattern over the weights of all transactions in the database. However, the weighted support does not satisfy the anti-monotone property. Yun *et al.* [45] then used the maximum weight of an item in the whole database as the weighted upper bound of each transaction. This upper-bound model has a downward-closure property. Later, the IUA algorithm [25] uses a tighter upper bound that uses the maximum weight of the transaction as the weighted upper bound of that transaction. Recently, a new upper-bound model based on IUA was proposed [23]. It used the highest possible weighted support in all possible extensions of the pattern of the weighted upper bound, which outperformed the upper bounds of IUA. In the real world, there are also specific kinds of rich data. For instance, incremental data [11, 35], uncertain data [28, 33], dynamic data [5, 22], and so on. Notice that we focus on weighted pattern mining in static data in this paper.

3 PRELIMINARIES AND PROBLEM FORMULATION

3.1 Problem statement

Let $\mathcal{D} = \{T_1, T_2, \ldots, T_n\}$ be a dataset and T_i is the transaction. The transaction T of size m can be denoted as $T = \{I_1, I_2, \ldots, I_m\}$, where I_i is the item in i-th position. Each sequence has its own binary index: $C \in \{c_0, c_1\}$. $G(p, T_i) = 1$ if the pattern p is contained in sequence T, otherwise it is 0. For each pattern p, $G(p, T_i)$ is evaluated for each T_i, $i = 1, \ldots, n$ and the following 2×2 contingency tables are constructed.

Table 1: The 2×2 contingency table

Variables	$G(P, T_i) = 1$	$G(P, T_i) = 0$	Row totals
$C = c_1$	a_p	$n_1 - a_p$	n_1
$C = c_0$	$x_p - a_p$	$n - n_1 + a_p - x_p$	$n - n_1$
Column totals	x_p	$n - x_p$	n

The total number of transactions is n. n_1 is the number of transactions with label $C = c_1$. x_P is the support of pattern P, i.e., the number of transactions that contain pattern p. a_s is the support of pattern s in class $C = c_1$, i.e., the number of transactions in class $C = c_1$ that contain the pattern p. From a statistical standpoint, our goal is to determine whether the observed value of a_p can represent that pattern p being over-represented in one of the two classes of C, implying that the occurrence of the pattern p and the class label c are dependent random variables. In the next part, the assessment is explained in a statistically precise manner.

3.2 Statistical association test

In the statistical association test, we are supposed to determine whether $G(p, T_i)$ and label C are significantly associated or not. The null hypothesis H_0 assumes that the two variables $G(p, T_i)$ and C are statistically independent, that is, no association. Fisher's exact test [9] is one of the most popular methods to test the null hypothesis. In this method, x_p, n_1, n in the 2×2 contingency tables are fixed. Under the null hypothesis, the probability of observing the current Table 1 follows a hypergeometric distribution:

$$\mathbf{P}(a_p = a \mid x_p, n_1, n) = \frac{\binom{n_1}{a}\binom{n - n_1}{x_p - a}}{\binom{n}{x_p}}$$

The p-value of observing the contingency table which is equal to or more extreme than the current table under the condition of null hypotheses is:

$$\mathbf{p}_p(u) = \sum_{k \mid \mathbf{P}(k \mid x_p, n_1, n) \le \mathbf{P}(u \mid x_p, n_1, n)} \mathbf{P}(k \mid x_p, n_1, n).$$

If $\mathbf{p}_p(u) \le \alpha$, we will reject the null hypotheses, that is, the pattern p is significant. It is worth noting that there is a trade-off for the value of the threshold α between Type I error (discover a false associate) and Type II error (miss a true associate). In the most common case, α is set as 0.05.

3.3 Multiple testing

When multiple hypotheses repeatedly test n times in the meanwhile, the expected number of false positives occurs is $\alpha \times n$, where α is the significant level. In this case, the probability that at least one false positive occurs will be much larger than α. To correct this problem, two criteria are widely used: the family-wise error rate (FWER for short) and false discovery rate (FDR for short) [2]. Here, we focus on family-wise error rate. FWER is the probability to occur at least one false positive. It is defined as

$$\text{FWER} = \mathbf{P}(\mathbf{FP} > 0).$$

Here **FP** means the number of false positives. In order to guarantee FWER is smaller than threshold α, $\mathbf{P} \le \alpha$ is replaced by $\mathbf{P} \le \delta$, where δ is called the corrected significance threshold. Since larger δ has more power to discover truly associated patterns, it is:

$$\delta^* = max\{\delta \mid \text{FWER}(\delta) \le \alpha\}.$$

One of the most popular methods for estimating FWER(δ) is the permutation testing proposed by Westfall and Young [44]. The idea is that by randomly labeling the class of the transactions, a new resampled dataset is generated. In the new dataset, no pattern p is truly associated with the resampled label. Thus, we can check whether false positives have occurred by calculating $\mathbf{p}_{min} = min_p \mathbf{p}_p$ and checking if $\mathbf{p}_{min} < \delta$. Both cases mean that at least one false positive occurred, which means FP > 0. Repeat this process J times[1], we have

$$\text{FWER}(\delta) = \frac{1}{J} \sum_{j=1}^{J} \mathbf{1}[\mathbf{p}_{min}^{(j)} \le \delta],$$

where $\mathbf{1}[\bullet]$ is an indicator function that takes the value 1 if its argument is true and 0 otherwise.

3.4 Weighted upper-bound

Islam *et al.* [23] use the Maximum Possible Weighted Support (MaxPWS for short) as the weighted upper-bound model to mine frequent patterns. The idea of MaxPWS is used as the highest possible weighted support value in all possible extensions of a pattern. Here, we modify it to adapt SSPM. The relevant definitions are stated in the following.

Definition 3.1. The weight value of a pattern p (W_p) is the sum of all the weights value of items in p over the number of items in p. That is

$$W_p = \frac{\sum_{i \in p}^{|p|} W_i}{|p|}.$$

*Definition 3.2 (**Local maximum weight**).* Local maximum weight, *LMW*, is the maximum weight value of an individual item in a sequence. The total sequential maximum weight, *tsmw*, is the sum of all *LMW* of the sequence contained in the database *DB*.

$$tsmw = \sum_{Seq \in DB} LMW(Seq).$$

For example, in the sequence *ABCEFD* in Table 2, where the weighted value of *A*, *B*, *C*, *D*, *E*, *F* are 0.5, 0.2, 0.7, 0.3, 0.1, and 0.25,

[1] J can be set to 10^3 or 10^4. In this paper, $J = 10^3$

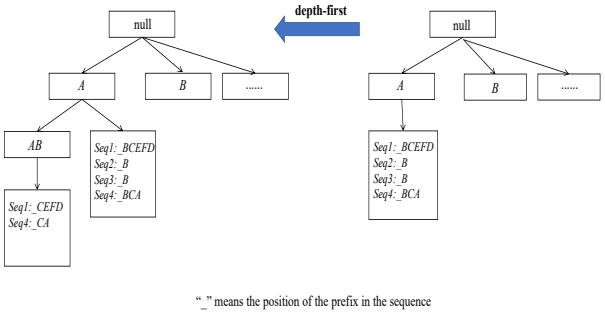

Figure 1: Depth-first mining process

respectively. In this case, the *LMW* of this sequence is 0.7 of item *C*. In Table 2, $tsmw = 0.7 + 0.5 + 0.5 + 0.7 = 2.4$.

Table 2: Four sequences in the example

ID	Sequence
Seq1	*A B C E F D*
Seq2	*A B*
Seq3	*A B*
Seq4	*A B C A*

*Definition 3.3 (**Weighted support**).* The weighted support, *WS*, is the value that measures whether a pattern is interesting. A pattern is considered to be interesting if its *WS* is larger than the predefined weighted support threshold θ. *WS* is defined as:

$$WS_P = \frac{\sum_{Seq:p \subseteq Seq} W_p}{tsmw}.$$

*Definition 3.4 (**Potential Extend Position**).* If *Supportset(p)* contains the ID of transaction that contains pattern p. The potential extend position of pattern p, $PEP_{+x}(p)$, is the subset of *SupportSet(p)*, in which the extension of p has x-more new positions may still occur.

For example, in Table 2 and Figure 1, the pattern A with its projected sequence *_BCEFD* with *Seq1* has five potential extend positions; the pattern A with its two projected sequence *_B* with *Seq2* and *Seq3* has one potential extend positions; the pattern A with its projected sequence *_BCA* with *Seq4* has three potential extend positions. That is, the number of depth-first mining potential extend positions is equal to the size of the project *seq*. Thus, in this case, $PEP_{+5}(A) = \{Seq1\}$, $PEP_{+1}(A) = \{Seq2, Seq3\}$, $PEP_{+3}(A) = \{Seq4\}$.

*Definition 3.5 (**Possible Weighted Support**).* Possible weighted support, *PWS*, is the potential-weighted support value of the pattern p. It is calculated by using a related *PEP* set. It is defined as:

$$PWS_{+x}(p) = \sum_{Seq:Seq \in PEP_{+x}(p)} \frac{|p| \times W_p + x \times LMW(Seq)}{|p| + x}.$$

The weighted upper-bound is *MaxPWS* that uses the highest possible weighted support among all PWS_{+x} to divide *tsmw*.

We still use the pattern A in Table 2 and Figure 1 as an example. $PWS_{+5} = \frac{1\times0.5+5\times0.7}{1+5} = \frac{2}{3}$, $PWS_{+1} = \frac{1\times0.5+1\times0.5}{1+1} + \frac{1\times0.5+1\times0.5}{1+1} = 1$, and $PWS_{+3} = \frac{1\times0.5+3\times0.7}{1+3} = 0.65$. The largest one is $PWS_{+1} = 1$. Therefore, $MaxPWS(p) = \frac{1}{2.4} = \frac{5}{12}$.

4 THE PROPOSED ALGORITHM

4.1 The minimum attainable p-value

To shrink the calculation of estimating FWER, the minimum attainable p-value is introduced [40]. Due to the 2×2 contingency Table 1 is discrete, the optional range of a_p is $a_p \in [a_{p,min}, a_{p,max}]$ where $a_{p,min} = max\{0, x_p - (n - n_1)\}$ and $a_{p,max} = min\{x_p, n_1\}$. The minimum attainable p-value $\Psi(x_p)$ which is strictly greater than 0 is

$$\Psi(x_p) = min\{\mathrm{p}_p(u) \mid a_{p,min} \le u \le a_{p,max}\}.$$

In Fisher's exact test, $\Psi(x_p)$ reaches its minimum attainable p-value when $a_p = a_{p,min}$ or $a_{p,max}$. Besides, a monotonically decreasing lower bound $\hat{\Psi}(x_p)$ of $\Psi(x_p)$ is introduced.

$$\hat{\Psi}(x_p) = \begin{cases} \Psi(x_p) & \text{if } 0 \le x_p \le n_1, \\ 1 \Big/ \binom{n}{n_1} & \text{if } n_1 < x_p \le n \end{cases}$$

In the multiple tests of our WSSpm algorithm, $\hat{\Psi}$ is used to control the value of FWER. The detail is stated in Subsection 4.4.

4.2 Weighted upper-bound model

To improve the efficiency of weighted upper-bound calculation, two hashmaps named *preWeight* and *preWeightFreq* are used [23]. *preWeight* stores the sum of *LMW* for those sequences that are contained in related PEP_{+x} and *preWeightFreq* stores the number of such sequences. Both the key of these two hashmaps is x. Take the pattern A in Table 2 as an example. From the sets of $PEP_{+x}(p)$ we can get *preWeight* (A) ={"5":0.7, "1":0.5+0.5, "3":0.7}, *preWeightFreq(A)* = {"5":1, "1":2, "3":1}. In our proposed algorithm, the pseudocode of operating these two haspmaps of new pattern extension is listed in Algorithm 1. First, a new pattern p' will be generated (line 1). Then, based on the projected database of prefix p, haspmap $PEP(p')$ will be built with key x and *SeqID* (line 3). For each key x in haspmap $PEP(p')$, WSSpm calculates $preWeight(p')$ and $preWeightFreq(p')$ (lines 4 to 7).

Algorithm 1: *pattern extend*

Input: p old pattern; i, extend item
1 $p' \leftarrow p.append(i)$;
2 $PEP(p')$;
3 **for** $x \in PEP(p')$ **do**
4 $\quad PreWeight(p')\,[x] += LMW.get(PEP(p').get(x))$;
5 $\quad preWeightFreq(p')\,[x] += 1$;
6 **end**

Before the calculation of weighted upper-bound *MaxPWS*, we check the weight support value by using a simple pruning check. That is *WScheck*.

*Definition 4.1 (**WScheck**).* Assume the maximum weight is denoted as MW. It is the largest individual item weight in the entire database and defined as

$$WScheck(p) = \frac{\sum_{seq:p\subseteq seq} MW}{tsmw}.$$

Hence, *MaxPws* is calculated when the value of *WScheck* is larger or equal to the weighted threshold θ.

The pseudocode of this weighted upper bound model is shown in Algorithm 2. This model needs *preWeight* and *preWeightFreq* of the new extended pattern p. From lines 1 to 4, we use the simple *WScheck*. If the value of *WScheck* is larger or equal to the weighted threshold, then we initialize *MaxPWS* in line 5. *leftSum* and *n* increase with the sum of *LMW* and frequency from hashmaps *preWeight* and *preWeightFreq*. These two values will be iterated from higher to lower value of the key x in hashmap *preWeight*. (lines 6 to 9). In the calculation of *PWS* (line 10), *n* is used to count the total frequency of p, and *leftSum* is used to calculate the potential extended weighted support. *MaxPWS* is the maximum *PWS* in the iteration (lines 11 to 14). After dividing *MaxPWS* with *tsmw*, *MaxPWS* will be compared with a weighted threshold to decide whether the new pattern p is interesting or not (lines 15 to 19).

Algorithm 2: upbdcheck

Input: p, candidate; *preWeight* of p; *preWeightFreq* of p
Output: the results of the pruning

1 calculate $WScheck(p)$;
2 **if** $WScheck < \theta$ **then**
3 | **return** 0
4 **end**
5 $MaxPWS \leftarrow 0$, $leftSum \leftarrow 0$, $n \leftarrow 0$;
6 $keys \leftarrow$ the descending order of $preWeight(p)$;
7 **while** $x \in keys$ **do**
8 | $leftSum \leftarrow leftSum + preWeight[x]$;
9 | $n \leftarrow n + preWeightFreq[x]$;
10 | $PWS_{+x}(p) \leftarrow \frac{n\times W_p\times|p|+x\times leftSum}{|p|+x}$;
11 | **if** $PWS > MaxPWS$ **then**
12 | | $MaxPWS \leftarrow PWS$;
13 | **end**
14 **end**
15 $MaxPWS \leftarrow \frac{MaxPWS}{tsmw}$;
16 **if** $MaxPWS < \theta$ **then**
17 | **return** 0
18 **end**
19 **return** 1

4.3 Proposed weighted support threshold

In multiple hypothesis tests, the significant threshold α, which is predefined to control the family-wise error rate (FWER), will decide the minimum frequency of interesting patterns. However, the traditional weighted support threshold is a user-defined constant that does not match the floating minimum frequent support threshold of multiple hypothesis tests. Thus, in this paper, we modify the minimum frequent threshold of WYlight [29], which can satisfy the need for a weighted support threshold. The modified weighted support threshold is:

$$\theta_p = \frac{\sigma \times W_p}{tsmw}.$$

Here W_p is the weighted value of the pattern p which needs to be checked. In the WSSpm algorithm, θ will be updated with the changing of FWER. That is, when FWER is larger than the pre-defined target FWER threshold, the minimum frequent support threshold σ will be updated and FWER will be reduced by model $\hat{\Psi}(\sigma)$. The weighted support threshold for pattern p, θ_p, will be renewed from σ. Here, we prove that this new weighted support threshold can discover the weighted statistically significant patterns in the situation of multiple tests.

Theorem 4.2. A pattern p can pass the weighted support threshold θ_p, if and only if the pattern p is statistically significant under target FWER (i.e., α).

Proof. If the pattern p passes the weighted support threshold θ_p, which means the weighted support of p, $WS_p = \frac{\sum_{Seq:p\subseteq Seq} W_p}{tsmw}$, is larger than or equal to θ_p. Thus, the frequency of p is not less than σ. Recall that $\Psi(x_p) = min\{\mathbf{p}_p(u) \mid a_{p,min} \le u \le a_{p,max}\}$, where $a_{p,min} = max\{0, x_p - (n - n_1)\}$ and $a_{p,max} = min\{x_p, n_1\}$. Hence, the lower bound of the minimum attainable p-value of pattern p and $\hat{\Psi}(\sigma_p)$ are smaller than δ. Hence, the minimum attainable p-value is smaller than δ. Due to FWER(δ) = $\frac{1}{J}\sum_{j=1}^{J} \mathbf{1}\left[\mathbf{p}_{min}^{(j)} \le \delta\right]$, the pattern p is a testable pattern at significant level δ.

If the pattern s is statistically significant at the significant level δ, it means that the $\mathbf{p}_{min}$ of s, $min_s\ \mathbf{p}_s$, is smaller than δ. Thus, the minimum attainable p-value of s is smaller than δ, $\Psi(\sigma_s) < \delta$. By the definition of $\Psi(\sigma_s)$, $\Psi(x_p) = min\{\mathbf{p}_p(u) \mid a_{p,min} \le u \le a_{p,max}\}$, the frequency of s, x_s is greater than σ. Hence, the weighted support value of s, WS_s is larger than the weighted support threshold θ_s. □

With the correctness of our weighted support threshold, we can prove the correctness of the weighted upper bound in SSPM. Islam [23] has proven that *MaxPWS* is anti-monotonic. *MaxPWS* is always larger than the weighted support value, *WS*, of the pattern p. Thus, we will show that using our weighted support threshold θ_p will not affect completeness.

Theorem 4.3. If $MaxPWS(p) < \theta_p$, then p is not statistically significant under the intended FWER, α, and pruning p will not have an impact on completeness.

Proof. $MaxPWS(p) < \theta_p$ means that the maximum weighted support value of potential pattern extension of p is smaller than θ_p. By the anti-monotonic of *MaxPWS*, we know that the weighted support value of all potential extensions of p is smaller than θ_p. Since WS_p is always smaller than $MaxPWS(p)$, we can get $WS_p < \theta_p$. Thus, the weighted support value of the pattern p and all its potential extensions are smaller than θ_p. Using Theorem 4.2, we know that p and its potential extension are not statistically significant under the target FWER, α. Besides, pruning p does not cause loss of objected supper-pattern of p, which ensures completeness. □

4.4 Proposed WSSpm algorithm

In this subsection, we introduce the details of the proposed WSSpm algorithm. In the main function 3 of WSSpm, it first preprocesses the database $\mathcal{D}$: calculate MW, $tsmw$ of $\mathcal{D}$, and LMW of each sequence in $\mathcal{D}$ (line 1). Then it precomputes all J permuted class label vector c and initialized all $\mathbf{p}_{min}^{j}$ as 1 (lines 2 to 5). Next, it sets the frequent support σ as 1 and uses the minimum attainable p-value $\hat{\Psi}$ for patterns with frequency σ to calculate the corrected significant threshold δ (line 6).

After this preprocessing, it begins the pattern enumeration procedure. This process is a kind of depth-first mining process, as shown in Figure 1. Line 7 begins the enumeration process from the root of the mining tree by calling the *ProcessNext* function. Note that for the dataset that has no root, we can define an empty set as a dummy root to keep generality.

In the *ProcessNext* method (Algorithm 4), it first computes all the possible p-value using the hypergeometric formula with fixed x_p, n_1, and n in line 1. These fixed parameters can reduce the computational complexity of Fisher's exact test. The result of the computation in line 13 can be shared across all J permutations. In lines 2 to 5, a_p for all permutations are calculated. The p-value, $\mathbf{p}_{a_p}^{j} = \mathbf{p}_p(a_p^{(j)})$, is updated if $\mathbf{p}_p^{j} < \mathbf{p}_{min}^{(j)}$. The current estimate of FWER with level δ is updated in line 6. It then checks whether the current estimate of FWER is larger than the target α. If it is, the current threshold δ will be decreased to lower than α by increasing σ (lines 7 to 10). From lines 11 to 16, it continues the pattern enumeration process. It will calculate weighted support θ_p in the current σ in line 12. It is a depth-first tree mining along with every child of pattern p that is weighted statistical frequent at its current modified weighted support θ_p or passes the weighted upper-bound test (line 13).

This depth-first enumeration process and weighted support threshold adjusting process will not stop unless the Westfall-Young method is finished. That is, the weighted support threshold θ at the end of the Westfall-Young method is the minimum-weighted support. After finishing the *ProcessNext* function, the main function 3 will calculate δ^* as the α-quantile of the set minimum p-value $\{\mathbf{p}_{min}^{(j)}\}_{j=1}^{J}$ from lines 8 to 9.

Algorithm 3: WSSpm

Input: $\mathcal{D}$, the original database; c, the class label; W, the weighted of each item in $\mathcal{D}$; B, the number of permutation; α the target FWER.
Output: P, all the weighted statistically significant patterns.

1 calculate MW, LMW and $tsmw$ of $\mathcal{D}$;
2 **for** $j = 1, \ldots, J$ **do**
3 $\quad c_j \leftarrow randperm(c)$;
4 $\quad \mathbf{p}_{min}^{j} \leftarrow 1$;
5 **end**
6 $\sigma \leftarrow 1, \delta \leftarrow \hat{\Psi}(\sigma)$;
7 call ProcessNext($root, n$);
8 $\mathbf{P}_{sort} \leftarrow sort(\{\mathbf{p}_{min}^{(j)}\}_{j=1}^{J})$;
9 $\delta^* \leftarrow max\{\mathbf{P}_{sort}^{(x)} \mid \mathbf{P}_{sort}^{(x)} < \mathbf{P}_{sort}^{(\alpha \times J+1)}\}$;
10 **return** c with p-value $< \delta^*$;

Algorithm 4: ProcessNext

Input: S pattern; X_s, the support of pattern s

1 Compute *p-value* $p_s(\mu)$ for all $\mu \in [a_{s,min}, a_{s,max}]$;
2 **for** $j \leftarrow 1$ *to* J **do**
3 $\quad$ Calculate $a_s^{(j)}$;
4 $\quad \mathbf{p}_{min}^{(j)} \leftarrow min\{\mathbf{p}_{min}^{(j)}, \mathbf{p}_s(a_s^{(j)})\}$;
5 **end**
6 FWER(δ) $\leftarrow \frac{1}{J}\sum_{j=1}^{J} 1[\mathbf{p}_{min}^{(j)} \le \delta]$;
7 **while** FWER(δ) $> \alpha$ **do**
8 $\quad \sigma \leftarrow \sigma + 1, \delta \leftarrow \hat{\Psi}(\sigma)$;
9 $\quad$ FWER(δ) $\leftarrow \frac{1}{J}\sum_{j=1}^{J} 1[\mathbf{p}_{min}^{(j)} \le \delta]$;
10 **end**
11 **for** $s' \in$ *Children(s)* **do**
12 $\quad \theta_{s'} \leftarrow \frac{\sigma \times W_{s'}}{tsmw}$;
13 $\quad$ **if** $WS_{s'} \ge \theta_{s'}$ *or upbdcheck*(s') **then**
14 $\quad\quad$ call ProcessNext($s', x_{s'}$);
15 $\quad$ **end**
16 **end**

4.5 Correctness of WSSpm

Here, we prove that our WSSpm algorithm can discover the weighted statically significant pattern correctly. It is equal to prove the following theorem.

Theorem 4.4. WSSpm returns the exact objected patterns to $\delta^* = max\{\delta | \text{FWER}(\delta) \le \alpha\}$ and related weighted support threshold θ_p, where δ^* and θ_p come from Sections 3.3 and 4.3, respectively.

From Ref. [29], we know that the Westfall-Young method has the following two properties. Firstly, the current estimated FWER will not decrease with the increase of objected pattern when δ is fixed. Secondly, FWER with $\delta < \hat{\Psi}(\sigma - 1)$ can be calculated using p-value of a pattern whose $x_p \ge \sigma$. Using these two properties, we can prove this theorem.

Proof. From the first property, we know that when FWER $> \alpha$, $\delta^* > \delta$. Thus, we should decrease FWER by reducing δ since FWER will keep increasing with the objected pattern processing. The second property can guarantee WSSpm can compute δ^* by using x_p. Let σ be the minimum frequent threshold and $\delta = \hat{\Psi}(\sigma)$ is related to significant level. $\delta_{old} = \hat{\Psi}(\sigma - 1)$ is its preceding significant level. In WSSpm, significant level will update when FWER(δ) $\le \alpha$, FWER(δ_{old}) $> \alpha$. Thus, $\delta^* \in [\delta, \delta_{old}]$. Here, we define $L(\delta) = \{p | \hat{\Psi}(x_p) \le \delta\}$. From Section 4.3, $WS_p \ge \theta_p \Leftrightarrow x_p \ge \sigma$. By the definition of $\hat{\Psi}(x_p)$, we have $x_p \ge \sigma \Leftrightarrow p \in L(\delta)$. Then, let $\mathbf{p}_{min}^{*} = min\{\mathbf{p}_p | p \in L(\delta)\}$. If $\mathbf{p}_{min}^{*} > \delta$, then $\mathbf{p}_p > \delta$ for $\forall p \in L(\delta)$, we can get $\forall p \notin L(\delta), \mathbf{p}_p \ge \hat{\Psi}(\sigma - 1) > \delta$. Thus, $\mathbf{1}[\mathbf{p}_{min} \le \alpha] = \mathbf{1}\left[\mathbf{p}_{min}^{*} \le \alpha\right]$. We can calculate all FWER(δ) $= \frac{1}{J}\sum_{j=1}^{J} \mathbf{1}\left[\mathbf{p}_{min}^{(j)} \le \delta\right]$ for all $\delta < \delta_{old}$. WSSpm will not calculate $\mathbf{p}_{min}$ that is larger than δ^*, increasing σ for pattern enumeration. Since $\delta^* < \delta_{old}$, the computed results δ^* of WSSpm are the weighted statistically significant patterns with FWER $< \alpha$. □

5 PERFORMANCE EVALUATION

In this section, we use four real-life datasets to evaluate the efficiency in execution time, maximum memory usage, number of tested patterns, and final minimum support of our proposed WSSpm algorithm. In an equal-weighted situation, we compare WSSpm with the WYlight algorithm [29], which is the state-of-the-art algorithm in the SSPM problem. In addition, to evaluate the influence of our weighted support threshold and weighted upper bound, we compare the performance of different versions of WSSpm. WSSpm is the version that uses *MaxPWS* weighted upper-bound. WSSpm$_{wc}$ is the version that only uses trivial *WScheck* weighted upper-bound. WSSpm* is the version that does not use the weighted upper bound.

5.1 Experimental setup and datasets

In experiments, all algorithms are written in Java and executed on a PC with an AMD Ryzen 7 4800H and 16 GB of memory. The operating system is Microsoft Windows 10 64-bit. The value of the pre-defined target α is set to 0.1, 0.05, 0.025, and 0.01 in the execution time and maximum memory usage tests. These values are frequently used in statistics. We set α at the most frequently used value of 0.05 in the analysis of the tested patterns and the final minimum support. The four experiments used real-world datasets of different kinds. The diverse characteristics of these datasets can represent the main types of data in real-life applications. The weight of each item is randomly generated, and its range is from 0 to 1. The characteristics of these four datasets are stated in the following.

- **Rosalind**$_{1836}$[2] is a chromosome with different fragments. This dataset has 50 transactions with 970 average lengths. The items are four types of nitrogen bases. The size of this dataset is 52 KB.
- **uniflna**$_{test}$[3] is the genetic fragment of a coronavirus. It has 100 transactions with an average length of transactions is 1616 items. The items are 11 types of items. The size of this dataset is 160 KB.
- **DRB**$_{10101}$ [4]is a sequential dataset of peptide fragments. This dataset has 100 transactions. The average length of the transaction is 15 items. It has 20 kinds of items and its size is 1.7 KB.
- > 0705172*A* [5]is a T cell evaluation dataset. It has 86 transactions with an average length of 972 items. It has 20 distinct items and its size is 93.4 KB.

5.2 Comparison with WYlight in equal weighted case

In this section, we compare the performance of our WSSpm algorithm and the performance of WYlight in an equal-weighted case. In these experiments, the weighted values of all items are set to 1. Two different versions of the WSSpm algorithm, WSSpm$_{WC}$, and WSSpm*, are also taken into compared experiments. We will analyze these algorithms in execution time, maximum memory usage, tested pattern, and final minimum support.

[2]https://github.com/jonhewz/DNASequences/blob/master/full_data_set.txt
[3]https://github.com/lukas-gust/data_mining_influenza/blob/master/uniflna_test1.csv
[4]http://www.cbs.dtu.dk/suppl/immunology/NetMHCIIpan-3.2/
[5]http://www.cbs.dtu.dk/suppl/immunology/NetMHCIIpan-3.2/

The result of execution time is represented in Figure 2. We can see that in these experiments, WSSpm and WYlight have similar execution times. In the three versions of WSSpm, WSSpm has the best performance, while WSSpm* has the longest execution time in most situations. However, the situation becomes different in the dataset DBR_{10101}. The running time of WSSpm is 25% faster than WYlight. WSSpm$_{WC}$ is 10% faster than WSSpm*. Maximum memory usage experiments in Figure 3 show that in most situations, although this difference is not obvious, WSSpm takes less memory than WYlight. However, in different versions of WSSpm, WSSpm$_{WC}$ takes less memory than WSSpm, while WSSpm* takes the largest memory usage.

The experiment results of tested patterns and final minimum support are shown in Table 4 and Table 3. We can see that in the experiments of the datasets $Rosalind_{1836}$, 0705172*A*, and $uniflna$ are the same. However, in the result of DBR_{10101}, the final minimum support of WSSPM is 25, while the final minimum support of WYlight, WSSpm*, and WSSpm$_{WC}$ is 24. The results of tested patterns show a similar situation. In the experiment of dataset DBR_{10101}, WSSpm has the smallest number of tested patterns, and the number of tested patterns of WYlight is slightly larger than that of WSSpm. The number of tested patterns of WSSpm* is 25% larger than the number of WSSpm. The number of WSSpm$_{WC}$ is 20% larger than that of WSSpm. The other three datasets, however, do not open a gap between these four algorithms.

From the results of these experiments, we can see that the patterns in the dataset can't be extended any further because of the minimum support limit, and WSSpm doesn't fully show its advantage in getting rid of patterns that are not interesting. Due to the calculation of its weighted upper bound, WSSpm consumes more memory. When the patterns get further extended, WSSpm will show its advantage in shrinking the pattern range. Using the upper bound, WSSpm can test fewer potential patterns. Although the calculation of the upper bound will consume memory and time, WSSpm is still faster due to the shrinking of the mining range. Compared to WYlight, WSSpm has the advantage of shrinking the mining range, although this advantage is not obvious. This is due to the fact that the weighted value of all items is 1, causing our weighted upper bound to lose its power in the local maximum weight, *LMW*. The only part of our weighted upper bound that can work in this case is the potential-extended position, *PEP*. In an equal-weighted situation, the weighted upper bound will cause an increase in the final minimum support. Because the weighted upper bound will consider patterns whose weighted support value is smaller than the weighted support threshold and whose weighted upper bound is larger than the threshold. However, the upper bound will prune these uninteresting patterns. Thus, it will not affect the final results. These experiments show that all versions of WSSpm can maintain their effectiveness in equal-weighted cases. Compared to WYlight, WSSpm can keep its efficiency even in equal-weight situations.

5.3 Effect of weighted upper-bound

In this section, we examine the performance of our weighted support threshold and modified weighted upper bound in a randomly weighted case. In all the weighted experiments, the weighted value of all items is randomly generated, ranging from 0 to 1. We will

Table 3: Final minimum support with α 0.05f

Algorithm	$Rosalind_{1836}$	0705172A	DBR_{10101}	uniflna
WSSpm (equal)	851	86	25	101
WYlight (equal)	851	86	24	101
WSSpm^* (equal)	851	86	24	101
WSSpm_{wc} (equal)	851	86	24	101
WSSpm	851	86	24	101
WSSpm^*	851	86	24	101
WSSpm_{wc}	851	86	24	101

Table 4: Tested patterns with α 0.05f

Algorithm	$Rosalind_{1836}$	0705172A	DBR_{10101}	uniflna
WSSpm (equal)	4	20	466	11
WYlight (equal)	4	20	477	11
WSSpm^* (equal)	4	20	591	11
WSSpm_{wc} (equal)	4	20	560	11
WSSpm	4	20	466	11
WSSpm^*	4	20	483	11
WSSpm_{wc}	4	20	489	11

analyze the performance of WSSpm, WSSpm*, and WSSpm$_{wc}$ in aspects of execution time, maximum memory usage, tested patterns, and final minimum support.

The results of execution time are represented in Figure 4. We can see that the result is similar to the equal weight result. WSSpm has the shortest execution time, while WSSpm* is 15% slower than WSSpm, and WSSpm$_{wc}$ is 10% slower than WSSpm. However, when the pre-defined target FWER is 0.01, the execution times of WSSpm, WSSpm*, and WSSpm$_{wc}$ become similar in the datasets $Rosalind_{1836}$ and DBR_{10101}. The result of maximum memory usage is shown in Figure 5. We can see that in most situations, the memory usage of WSSpm is 20% larger than that of the other two algorithms, while the memory usage of WSSpm* and WSSpm$_{wc}$ is similar. The result of the final minimum support of the weighted case is shown in Table 3. In a weighted situation, all three versions of WSSpm have the same final minimum support value. The result of the tested pattern in a weighted case is represented in Table 4. We can see that the outcome is similar to the result in an equal-weighted case. The gap between these three algorithms is opened in the dataset DBR_{10101}. WSSpm has the smallest number of tested patterns. The number of tested patterns of WSSpm* is about 5% larger than the number of WSSpm, while the number of WSSpm$_{wc}$ is only slightly smaller than the number of WSSpm*.

From these results, we can see that even in the dataset where a pattern can not be further extended due to the limit of the pre-defined target FWER, WSSpm can still prune uninteresting patterns. This is the reason for the efficiency of WSSpm. This advantage is maintained when the pattern extends. The calculation of the weighted upper bound will consume time and memory, but the shrunken mining range can still decrease execution time. The final minimum support value of the three versions of WSSpm is equal to the equal-weighted experiment result of WYlight. The weighted value of an item does not affect the calculation of the family-wise error. This outcome shows that all WSSpm algorithms are correct in controlling FWER. However, the weighted upper bound can not prune uninteresting patterns efficiently, which makes the gap in the tested pattern between WSSpm and WSSpm* inconspicuous. Therefore, we need to improve the weighted upper bound in future studies.

6 CONCLUSION

In this paper, we introduce the problem of mining weighted statistically significant patterns and propose the WSSpm algorithm. We also propose a weighted support threshold that can guarantee correctness in SSPM and prove this correctness. By combining permutation testing and a modified weighted upper bound, the WSSpm algorithm can efficiently mine weighted statistically significant patterns. Using properties of permutation tests, we prove the effectiveness of WSSpm theoretically. Several experiments have shown that, compared to the related state-of-the-art algorithm, WSSpm can maintain its efficiency in execution time and memory usage in an equal-weighted case. Furthermore, our weighted support threshold and modified weighted upper bound can effectively shrink the mining range in a weighted situation.

In the future, we will investigate the relationship between weighted values and multiple hypothesis tests. We also want to test a weighted support threshold that works well with the general multiple hypothesis error-controlling process. Furthermore, we will look at the issue of the relationship between the weighted value and the p-value of the hypothesis test in multiple hypothesis cases.

ACKNOWLEDGMENTS

This research was supported in part by Guangzhou Basic and Applied Basic Research Foundation (No. 202102020277), Fundamental Research Funds for the Central Universities of Jinan University (No. 21622416), National Natural Science Foundation of China (Nos. 62002136 and 62272196), Natural Science Foundation of Guangdong Province (No. 2022A1515011861), the Young Scholar Program of Pazhou Lab (No. PZL2021KF0023), and Guangdong Key Laboratory for Data Security and Privacy Preserving. Dr. Wensheng Gan is the corresponding author of this paper.

REFERENCES

[1] Rakesh Agrawal, Tomasz Imieliński, and Arun Swami. 1993. Mining association rules between sets of items in large databases. In *The ACM SIGMOD International Conference on Management of Data*. ACM, 207–216.

[2] Yoav Benjamini and Daniel Yekutieli. 2001. The control of the false discovery rate in multiple testing under dependency. *Annals of Statistics* 29, 4 (2001), 1165–1188.

[3] Carlo 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.

[4] Chien-Ming Chen, Lili Chen, Wensheng Gan, Lina Qiu, and Weiping Ding. 2021. Discovering high utility-occupancy patterns from uncertain data. *Information Sciences* 546 (2021), 1208–1229.

[5] Pilsun Choi and Buhyun Hwang. 2017. Dynamic weighted sequential pattern mining for USN system. In *The 11th International Conference on Ubiquitous Information Management and Communication*. ACM, 1–6.

[6] EunYi Chung and Joseph P Romano. 2013. Exact and asymptotically robust permutation tests. *Annals of Statistics* 41, 2 (2013), 484–507.

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

[8] Wouter Duivesteijn and Arno Knobbe. 2011. Exploiting false discoveries–statistical validation of patterns and quality measures in subgroup discovery. In *The IEEE 11th International Conference on Data Mining*. IEEE, 151–160.

[9] 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.

[10] Philippe Fournier-Viger, Wensheng Gan, Youxi Wu, Mourad Nouioua, Wei Song, Tin Truong, and Hai Duong. 2022. Pattern mining: Current challenges and opportunities. In *International Conference Database Systems for Advanced Applications International Workshops*. Springer, 34–49.

[11] Wensheng Gan, Jerry Chun-Wei Lin, Philippe Fournier-Viger, Han-Chieh Chao, Tzung-Pei Hong, and Hamido Fujita. 2018. A survey of incremental high-utility itemset mining. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 8, 2 (2018), e1242.

[12] Wensheng Gan, Jerry Chun-Wei Lin, Philippe Fournier-Viger, Han-Chieh Chao, Vincent S Tseng, and Philip S Yu. 2021. A survey of utility-oriented pattern mining. *IEEE Transactions on Knowledge and Data Engineering* 33, 4 (2021), 1306–1327.

[13] Wensheng Gan, Jerry Chun-Wei Lin, Philippe Fournier-Viger, Han-Chieh Chao, and Philip S Yu. 2019. A survey of parallel sequential pattern mining. *ACM Transactions on Knowledge Discovery from Data* 13, 3 (2019), 1–34.

[14] Wensheng Gan, Jerry Chun-Wei Lin, Philippe Fournier-Viger, Han-Chieh Chao, and Justin Zhan. 2017. Mining of frequent patterns with multiple minimum supports. *Engineering Applications of Artificial Intelligence* 60 (2017), 83–96.

[15] Wensheng Gan, Jerry Chun Wei Lin, Philippe Fournier-Viger, Han Chieh Chao, Justin Zhan, and Ji Zhang. 2018. Exploiting highly qualified pattern with frequency and weight occupancy. *Knowledge and Information Systems* 56, 1 (2018), 165–196.

[16] Wensheng Gan, Jerry Chun-Wei Lin, Jiexiong Zhang, Philippe Fournier-Viger, Han-Chieh Chao, and Philip S Yu. 2021. Fast utility mining on sequence data. *IEEE Transactions on Cybernetics* 51, 2 (2021), 487–500.

[17] Aristides Gionis, Heikki Mannila, Taneli Mielikäinen, and Panayiotis Tsaparas. 2007. Assessing data mining results via swap randomization. *ACM Transactions on Knowledge Discovery from Data* 1, 3 (2007), 14–24.

[18] Yijie Gui, Wensheng Gan, Yao Chen, and Yongdong Wu. 2022. Mining with Rarity for Web Intelligence. In *Companion Proceedings of the Web Conference*. ACM, 973–981.

[19] Wilhelmiina Hämäläinen. 2012. Kingfisher: an efficient algorithm for searching for both positive and negative dependency rules with statistical significance measures. *Knowledge and Information Systems* 32 (2012), 383–414.

[20] Wilhelmiina Hämäläinen and Geoffrey I Webb. 2019. A tutorial on statistically sound pattern discovery. *Data Mining and Knowledge Discovery* 33, 2 (2019), 325–377.

[21] Jiawei Han, Jian Pei, and Yiwen Yin. 2000. Mining frequent patterns without candidate generation. *ACM SIGMOD Record* 29, 2 (2000), 1–12.

[22] Sabrina Zaman Ishita, Faria Noor, and Chowdhury Farhan Ahmed. 2018. An efficient approach for mining weighted sequential patterns in dynamic databases. In *The Industrial Conference on Data Mining*. Springer, 215–229.

[23] Md Ashraful Islam, Mahfuzur Rahman Rafi, Al-amin Azad, and Jesan Ahammed Ovi. 2021. Weighted frequent sequential pattern mining. *Applied Intelligence* 52, 1 (2021), 1–28.

[24] Junpei Komiyama, Masakazu Ishihata, Hiroki Arimura, Takashi Nishibayashi, and Shin-ichi Minato. 2017. Statistical emerging pattern mining with multiple testing correction. In *The 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 897–906.

[25] Guo-Cheng Lan, Tzung-Pei Hong, and Hong-Yu Lee. 2014. An efficient approach for finding weighted sequential patterns from sequence databases. *Applied Intelligence* 41, 2 (2014), 439–452.

[26] Duy Nguyen Le Vo, Takuto Sakuma, Taiju Ishiyama, Hiroki Toda, Kazuya Arai, Masayuki Karasuyama, Yuta Okubo, Masayuki Sunaga, Hiroyuki Hanada, and Yasuo Tabei. 2020. Stat-DSM: Statistically discriminative sub-trajectory mining with multiple testing correction. *IEEE Transactions on Knowledge and Data Engineering* 34, 3 (2020), 1477–1488.

[27] Jiuyong Li, Jixue Liu, Hannu Toivonen, Kenji Satou, Youqiang Sun, and Bingyu Sun. 2014. Discovering statistically non-redundant subgroups. *Knowledge-Based Systems* 67 (2014), 315–327.

[28] Jerry Chun-Wei Lin, Wensheng Gan, Philippe Fournier-Viger, Tzung-Pei Hong, and Vincent S Tseng. 2016. Efficient algorithms for mining high-utility itemsets in uncertain databases. *Knowledge-Based Systems* 96 (2016), 171–187.

[29] Felipe Llinares-López, Mahito Sugiyama, Laetitia Papaxanthos, and Karsten Borgwardt. 2015. Fast and memory-efficient significant pattern mining via permutation testing. In *The 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 725–734.

[30] 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 *The European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 422–436.

[31] Leonardo Pellegrina, Matteo Riondato, and Fabio Vandin. 2019. SPuManTE: Significant pattern mining with unconditional testing. In *The 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 1528–1538.

[32] Leonardo Pellegrina and Fabio Vandin. 2018. Efficient mining of the most significant patterns with permutation testing. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2070–2079.

[33] Md Mahmudur Rahman, Chowdhury Farhan Ahmed, and Carson Kai-Sang Leung. 2019. Mining weighted frequent sequences in uncertain databases. *Information Sciences* 479 (2019), 76–100.

[34] GD Ramkumar, Sanjay Ranka, and Shalom Tsur. 1998. Weighted association rules: Model and algorithm. In *The 4th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 661–666.

[35] Kashob Kumar Roy, Md Hasibul Haque Moon, Md Mahmudur Rahman, Chowdhury Farhan Ahmed, and Carson Kai-Sang Leung. 2022. Mining weighted sequential patterns in incremental uncertain databases. *Information Sciences* 582 (2022), 865–896.

[36] Huijun Tang, Jiangbo Qian, Yangguang Liu, and Xiao-Zhi Gao. 2022. Mining statistically significant patterns with high utility. *International Journal of Computational Intelligence Systems* 15, 1 (2022), 1–19.

[37] Feng Tao, Fionn Murtagh, and Mohsen Farid. 2003. Weighted association rule mining using weighted support and significance framework. In *The 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 661–666.

[38] Robert E Tarone. 1990. A modified Bonferroni method for discrete data. *Biometrics* 46, 2 (1990), 515–522.

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

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

[41] Thien Q Tran, Kazuto Fukuchi, Youhei Akimoto, and Jun Sakuma. 2020. Statistically significant pattern mining with ordinal utility. In *The 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 1645–1655.

[42] Shicheng Wan, Jiahui Chen, Peifeng Zhang, Wensheng Gan, and Tianlong Gu. 2022. Discovering top-k profitable patterns for smart manufacturing. In *Companion Proceedings of the Web Conference*. ACM, 956–964.

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

[44] Peter H Westfall and S Stanley Young. 1993. *Resampling-based multiple testing: Examples and methods for p-value adjustment*. Vol. 279. John Wiley & Sons.

[45] Unil Yun and John J Leggett. 2005. WFIM: Weighted frequent itemset mining with a weight range and a minimum weight. In *The 15th SIAM International Conference on Data Mining*. Society for Industrial and Applied Mathematics, 636–640.

[46] Unil Yun and John J Leggett. 2006. WSpan: Weighted sequential pattern mining in large sequence databases. In *The 3Rd International IEEE Conference Intelligent Systems*. IEEE, 512–517.

[47] Chunkai Zhang, Zilin Du, Yuting Yang, Wensheng Gan, and Philip S Yu. 2021. On-shelf utility mining of sequence data. *ACM Transactions on Knowledge Discovery from Data* 16, 2 (2021), 1–31.

A APPENDIX

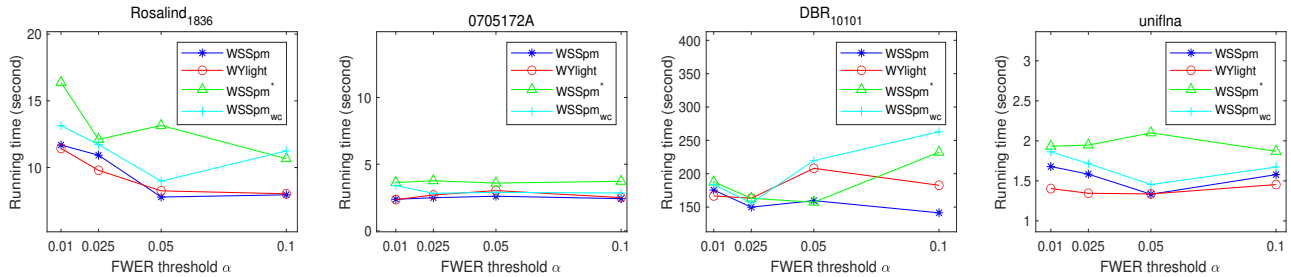

Figure 2: Execution time in an equal weighted case.

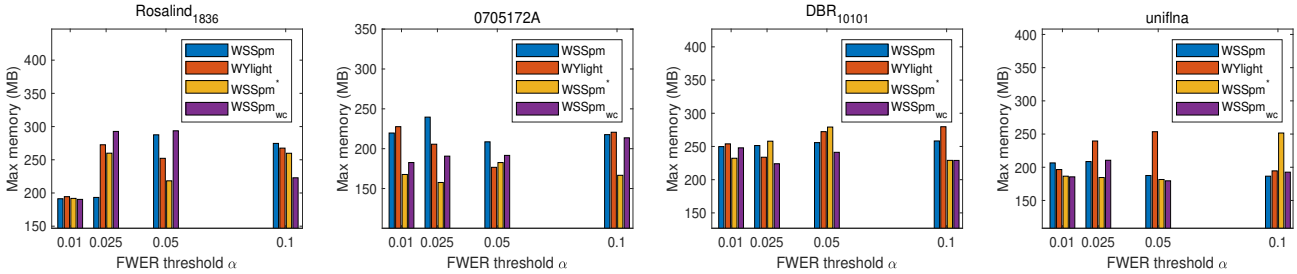

Figure 3: Memory usage in an equal weighted case.

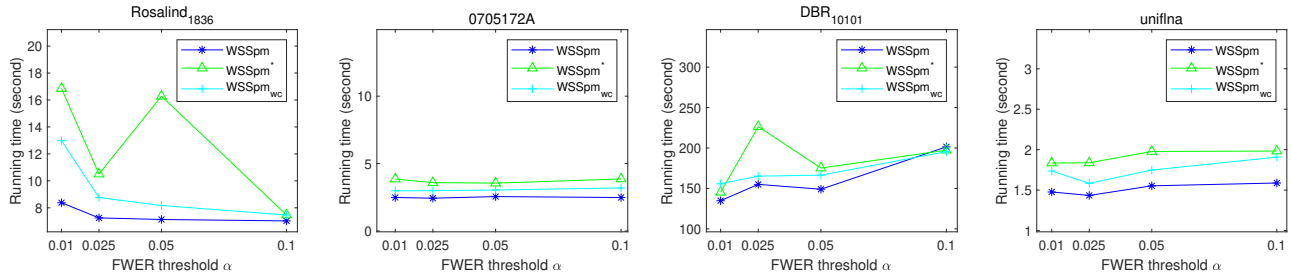

Figure 4: Execution time in a random weighted case.

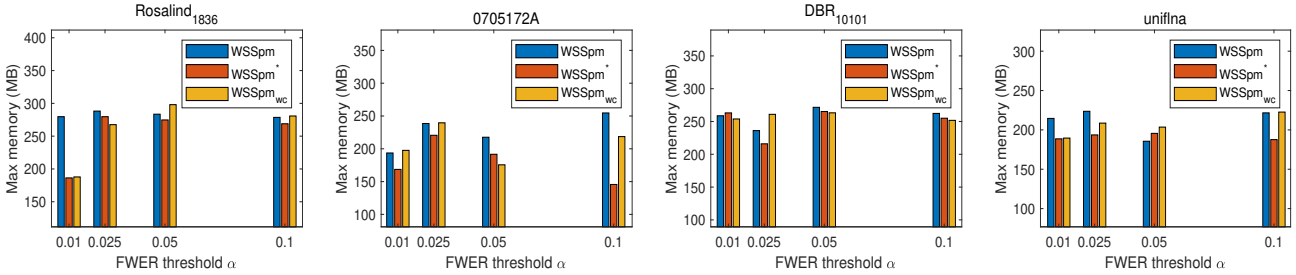

Figure 5: Memory usage in a random weighted case.