Permutation Strategies for Mining Significant Sequential Patterns

Andrea Tonon
Department of Information Engineering
Università di Padova
Padova, Italy
andrea.tonon@dei.unipd.it

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

***Abstract*—The identification of significant patterns, defined as patterns whose frequency significantly deviates from what is expected under a suitable null model of the data, is a key data mining task with application in several areas. We present PROMISE, an algorithm for identifying significant sequential patterns while guaranteeing that the probability that one or more false discoveries are reported in output (i.e., the Family-Wise Error Rate - FWER) is less than a user-defined threshold. PROMISE employs the Westfall-Young method to correct for multiple hypothesis testing, a more powerful method than the commonly used Bonferroni correction. PROMISE crucially hinges on the generation of (random) permuted datasets with features similar to the input dataset, for which we provide two efficient strategies. We also provide a rigorous analysis of one of such strategies, which is based on a properly defined swap operation, proving a rigorous bound on the number of swaps it requires. The results of our experimental evaluation show that PROMISE is an efficient method that allows the discovery of statistically significant sequential patterns from transactional datasets while properly controlling for false discoveries.**

***Index Terms*—Sequential Patterns; Statistical Pattern Mining; Hypothesis Testing; Permutation Test.**

I. INTRODUCTION

Sequential pattern mining [1] is a fundamental data mining task that finds application in several areas. In its original formulation [1], it requires to extract all sequential patterns, defined as sequences of itemsets, that appear in a fraction at least θ of the transactions of a transactional sequential dataset, where each transaction is a sequence of itemsets. Since it formulation, several methods (e.g., [2]–[5]) have been designed to efficiently extract sequential patterns from a dataset.

While the frequency of a pattern is an important feature in some applications, it is usually not sufficient to identify patterns that provide useful knowledge regarding the process described by the data. For example, a sequence of itemsets may appear frequently in a dataset simply because each of the itemsets has high individual frequency, even if there is no association between the itemsets in the sequence. A natural framework to identify interesting sequential patterns is provided by *statistical hypothesis testing*, where the goal is to mine *statistically significant sequential patterns*, defined as sequences that appear *more frequently than expected* under an appropriate (generative) *null model* for the data.

The extraction of statistically significant patterns has received a lot of attention when patterns are itemsets, with several methods [6]–[9] that have been proposed to identify significant itemsets while providing guarantees on false discoveries (i.e., itemsets flagged as significant but that are not). The most commonly used guarantee is given by the Family-Wise Error Rate (FWER), that is the probability that one or more false discovery is reported in ouput. Strikingly, only few methods [10], [11] to identify statistically significant *sequential* patterns have been proposed. The extraction of statistically significant sequential patterns is more complex than the extraction of significant itemsets, mostly for two key issues: first, the number of sequential patterns that can be built from a ground set of items is much larger than the number of itemsets, and the gargantuan number of candidate sequential patterns poses a *multiple hypothesis testing* issue, since the probability of a false discovery increases with the number of candidate patterns; second, the definition of appropriate null models for sequential patterns is more difficult, since reasonable null models do not result in distributions that can be analytically described and it is therefore crucial to be able to efficiently compute the statistical significance of patterns. To the best of our knowledge, no method to rigorously identify statistically significant sequential patterns with rigorous guarantees on the FWER for the reported patterns is available (see also Section I-B).

A. Our Contribution

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

- We introduce a new algorithm, PROMISE, to identify statistically significant sequential patterns using permutation testing. PROMISE is the first algorithm to provide rigorous guarantees on the Family-Wise Error Rate (FWER) of the output, using the Westfall-Young method to properly correct for multiple hypothesis testing.
- We introduce and formalize two strategies, one based on itemsets swapping and one on permuting transactions, to generate (random) permuted datasets for sequential patterns mining. These two strategies are at the core of PROMISE, and they sample datasets from the distribution

of all datasets where the number of appearances of each itemset and the number of itemsets in each transaction are the same as in the input dataset.
- We provide a formal analysis of the itemsets swapping strategy, proving that a polynomial number of swaps are sufficient to uniformly sample a random dataset from the aforementioned distribution. Moreover, we experimentally show that a number of swaps proportional to the number of itemsets in the dataset is sufficient to sample a random dataset.
- We provide a parallel implementation of our algorithm and conduct an extensive experimental evaluation of its use to extract significant sequential patterns, showing that PROMISE allows to efficiently extract significant patterns from real sequential datasets.

B. Related Work

Since the introduction of the frequent sequential pattern mining problem [1], a number of methods have been developed to efficiently extract all frequent sequential patterns (see also [12] for several references). We focus on the identification of statistically significant sequential patterns and our algorithm PROMISE can employ any algorithm to extract frequent patterns from a (real or random) dataset as a subroutine.

Several works have been proposed to identify statistically significant itemsets where the significance is defined in terms of the comparison of itemsets' statistics (e.g., itemsets' frequencies or number) with a null mode (e.g., [6], [8]). The one that is most related to ours is the work of Gionis et al. [8], which introduces a swap randomization approach to assess the significance of patterns (e.g., itemsets) in 0-1 datasets. Such technique cannot be applied to sequential transactions, due to sequential dimension that is absent in 0-1 datasets. In addition, [8] provides only an experimental assessment of the number of swaps required to sample a random dataset, while we also prove an upper bound.

Few methods [10], [11] have been proposed to mine statistically significant sequential patterns. Gwadera and Crestani [10] proposes a method based on a null model obtained by combining two models at different levels, in particular itemset-wise and sequence-wise, with a maximum entropy model for the itemset-wise level and a mixture model for the sequence-wise level. Our null model is much simpler and allows for the efficient mining of significant sequential patterns, that is instead not possible with the method of [10], in particular for large patterns. More recently, Low-Kam et al. [11] introduce an approach based on the *independence model*, where each itemset appears in a transaction with probability equal to its frequency in the real dataset and independently of all other events. We consider a null model where transaction lengths are preserved as well, which is more appropriate in cases where different groups of transactions are in the dataset (similarly to what happens for itemsets [8]). In addition, [11] performs a Bonferroni correction assuming that candidates sequential patterns are only the frequent patterns *observed* in the dataset, while all potential candidate patterns should be considered, as it has been argued for other pattern mining problems [6].

A different line of works identifies significant patterns, including sequential patterns, where the significance is given by the association of the presence of the pattern with a binary label available from each transaction [13]–[16]. These methods cannot be applied to identify patterns whose frequency significantly deviates from a null model. Several methods (e.g., [17]–[19]) have been proposed to identify *interesting patterns* using some alternative interestingness measure. These measures, and the methods that employ them, are orthogonal to our approach, which focuses on the statistical significance of patterns.

II. PRELIMINARIES

We now provide the definitions and concepts used throughout the paper.

A. Sequential Pattern Mining

Let $\mathcal{I} = \{i_1, i_2, \ldots, i_h\}$ be a finite *ground set* of elements called *items*. An *itemset* S is a (nonempty) subset of $\mathcal{I}$, i.e. $S \subseteq \mathcal{I}$. A *sequence* $\mathbf{s} = \langle S_1, S_2, \ldots, S_\ell \rangle$ is a *finite ordered list of itemsets*, with $S_i \subseteq I, 1 \leq i \leq \ell$. The *length* $|\mathbf{s}|$ of $\mathbf{s}$ is defined as the number of itemsets in $\mathbf{s}$. The *item-length* $||\mathbf{s}||$ of $\mathbf{s}$ is the sum of the sizes of the itemsets in $\mathbf{s}$, i.e., $||\mathbf{s}|| = \sum_{i=1}^{|\mathbf{s}|} |S_i|$, where $|S_i|$ is the cardinality of itemset S_i. A sequence $\mathbf{a} = \langle A_1, A_2, \ldots, A_y \rangle$ is a *subsequence* of another sequence $\mathbf{b} = \langle B_1, B_2, \ldots, B_w \rangle$, denoted $\mathbf{a} \sqsubseteq \mathbf{b}$, if and only if there exist integers $1 \leq i_1 < i_2 < \ldots < i_y \leq w$ such that $A_1 \subseteq B_{i_1}$, $A_2 \subseteq B_{i_2}, \ldots, A_y \subseteq B_{i_y}$. Note that an item can occur only once in an itemset, but it can occur multiple times in different itemsets of the same sequence. A *dataset* D is a finite bag of *(sequential) transactions*, $D = \{\tau_1, \tau_2, \ldots, \tau_{|D|}\}$, where each transaction $\tau_i \in D$ is a sequence with items from the ground set $\mathcal{I}$. A sequence $\mathbf{s}$ *belongs* to a transaction $\tau \in D$ if and only if $\mathbf{s} \sqsubseteq \tau$. For any sequence $\mathbf{s}$, the *support set* $T_D(\mathbf{s})$ of $\mathbf{s}$ in D is the set of transactions in D to which $\mathbf{s}$ belongs: $T_D(\mathbf{s}) = \{\tau \in D : \mathbf{s} \sqsubseteq \tau\}$. The *support* $s_D(\mathbf{s})$ of $\mathbf{s}$ in D is the cardinality of the set $T_D(\mathbf{s})$, that is the number of transactions in D to which $\mathbf{s}$ belongs: $s_D(\mathbf{s}) = |T_D(\mathbf{s})|$. Finally the *frequency* $f_D(\mathbf{s})$ of $\mathbf{s}$ in D is the *fraction* of transactions in D to which $\mathbf{s}$ belongs: $f_D(\mathbf{s}) = \frac{s_D(\mathbf{s})}{|D|}$.

Example. Consider the dataset $D = \{\tau_1, \tau_2, \tau_3, \tau_4\}$ where:

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

D has 4 transactions. τ_1 has length $|\tau_1| = 3$ and item-length $||\tau_1|| = 4$. The frequency $f_D(\langle\{7\}\rangle)$ of $\langle\{7\}\rangle$ in D is 3/4, since $\langle\{7\}\rangle$ is contained in all the transactions but not in the second one. Note that while $\langle\{7\}\rangle$ occurs twice as subsequence of τ_1, τ_1 contributes only once to the frequency of $\langle\{7\}\rangle$. $\langle\{7\},\{6\},\{5\}\rangle$ is not a subsequence of τ_1 because the order of the itemsets in the two sequences is not the same.

Let $\mathbb{S}$ denote the set of all sequences built with itemsets containing items from $\mathcal{I}$. (Note that $\mathbb{S}$ is an infinite set.) Given

a dataset D and a *minimum frequency threshold* $\theta \in (0, 1]$, the *sequential pattern mining* task requires to output all sequences from $\mathbb{S}$ whose frequency in D is greater than θ, and their frequencies:

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

B. Significant Sequential Pattern Mining

The task of *mining significant sequential patterns* requires to identify sequential patterns whose frequency is *significant*, that is, whose frequency is not due to random fluctuations in the data. To assess the significance of a pattern, the framework of *statistical hypothesis testing* is usually employed. For each sequential pattern $\mathbf{s}$, let $H_{\mathbf{s}}$ be the *null hypothesis* that the frequency $f_D(\mathbf{s})$ of $\mathbf{s}$ in D well conforms to the frequency of $\mathbf{s}$ in a *random dataset*, i.e. a dataset taken uniformly at random among all datasets with properties similar to D. In this work the properties of interest are i) the number of times each itemset appears in D; ii) the length of each transaction in D. Therefore, under the null hypothesis D is a dataset taken uniformly at random from all datasets where each itemset appears the same number of times as in D *and* each transaction has the same length as in D. As an example for why preserving these properties in a random dataset is important, consider a sequential dataset from an e-commerce website. Each transaction is the ordered list of purchases made by a single user. The itemsets represent products bought together. In such a dataset the idea is to preserve the products that the users bought together (the itemsets and the number of times they appear) and also to preserve the number of orders made by the users (the length of the transactions).

Example. Consider the dataset D in Section II-A. A random dataset $\tilde{D}$ with the same properties of D is the following:

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

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

Under the null hypothesis the frequency of pattern $\mathbf{s}$ is described by a random variable $X_{\mathbf{s}}$. In order to assess the significance of $\mathbf{s}$, a *p-value* $p_{\mathbf{s}}$ is commonly computed. The *p-value* $p_{\mathbf{s}}$ is the probability of observing a frequency at least as large as the frequency $f_D(\mathbf{s})$ of $\mathbf{s}$ in D under the null hypothesis:

$$p_{\mathbf{s}} = \Pr[X_{\mathbf{s}} \geq f_D(\mathbf{s}) | H_{\mathbf{s}}].$$

C. Multiple-Hypothesis Testing

The statistical hypothesis testing framework is commonly used to provide guarantees on the *false discoveries*, i.e., patterns flagged as significant while they are not. When a single pattern $\mathbf{s}$ is tested for significance, flagging $\mathbf{s}$ as significant (i.e., *rejecting* the null hypothesis) when $p_{\mathbf{s}} \leq \alpha$, where α is a threshold fixed by the user, guarantees that the probability that $\mathbf{s}$ corresponds to a *false discovery* (i.e., is flagged as significant when it is not) is at most α.

The situation is completely different when several patterns are tested simultaneously, as in the case of pattern mining. If d patterns are tested with the approach used for a single pattern (i.e., each pattern is flagged as significant if its p-value is $\leq \alpha$), then the expected number of false discoveries can be as large as αd. To solve this issue one identifies a *corrected* threshold δ such that all patterns with p-value $\leq \delta$ can be reported while providing some guarantees on the number of false discoveries.

A common approach is to identify δ that provides guarantees on the *Family-Wise Error Rate* (FWER), defined as the probability of reporting at least one false positive. That is, if FP is the number of false positives, then FWER $= \Pr[FP > 0]$. For a given value δ, let FWER(δ) be the FWER obtained when δ is used as corrected significance threshold, that is by rejecting (i.e., flagging as significant) all null hypotheses (i.e., patterns) with p-value $\leq \delta$. Often FWER(δ) cannot be evaluated in closed form.

One approach to set δ is to use the *Bonferroni correction*, setting δ to δ/d. It is easy to show (using the union bound) that the resulting FWER satisfies FWER(δ) $\leq d\delta = \alpha$. However when d is large, as in the case of pattern mining, δ is very close to 0, resulting in low *statistical power* with many *false negatives* (i.e., significant patterns that are not reported in output). Note that this issue is particular severe for sequential patterns, since if one does not restrict (before analyzing the dataset) the space of patterns, i.e. hypotheses, the number of candidate patterns is *infinite*, and therefore $d = \infty$. Even restricting the set of patterns, for example considering only patterns of length at most ℓ, may result in low statistical power, since the number of candidate patterns increases exponentially with ℓ.

More sophisticated techniques have been designed to increase the statistical power. In particular, the Westfall-Young (WY) method [20] is a multiple hypothesis testing procedure based on permutation testing that results in high statistical power [15] and has been successfully applied in other pattern mining scenarios [15], [16]. The WY method directly estimates the *joint distribution* of null hypotheses using *permuted datasets*, i.e., datasets obtained from the distribution described by the null hypothesis. In detail, the WY method considers P permuted datasets $D_1, \ldots, D_P$ obtained uniformly at random among the space of all possible datasets (under the null hypothesis). Then for every dataset $D_i, i = 1, \ldots, P$, computes the minimum p-value $p^{(i)}_{\min}$ over all patterns (hypotheses) of interest. The FWER FWER(δ) resulting from using δ as

corrected significance threshold can then be estimated as:

$$\text{FWER}(\delta) = \frac{1}{P}\sum_{i=1}^{P} \mathbb{1}[p_{\min}^{(i)} \leq \delta]$$

where $\mathbb{1}[\cdot]$ is the indicator function (of value 1 if the argument is true, and 0 otherwise). Given a FWER threshold α, the corrected threshold δ^* is then obtained as

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

D. Estimating the p-value

Note that the statistical hypothesis testing framework requires to be able to compute the p-value $p_{\mathbf{s}}$. For complex null hypotheses, such as ours, the p-value cannot be computed analytically. However, when one can sample datasets uniformly at random from the distribution described by the null hypothesis, the p-value can be estimated by a simple Monte Carlo procedure as follows: sample T random datasets $D_1, \ldots, D_T$; for each dataset $D_i, i = 1, \ldots, T$, compute the frequency $f_{D_i}(\mathbf{s})$ of $\mathbf{s}$ in D_i; then the p-value $p_{\mathbf{s}}$ is estimated as

$$p_{\mathbf{s}} = \frac{1}{T+1}\left(1 + \sum_{i=1}^{T} \mathbb{1}[f_{D_i}(\mathbf{s}) \geq f_D(\mathbf{s})]\right).$$

III. ProMiSe: Mining Significant Sequential Patterns with Permutation Testing

In this section we describe ProMiSe, our algorithm to mine significant sequential patterns. ProMiSe is described in Algorithm 1. Given a sequential dataset D, a minimum frequency threshold $\theta \in (0, 1]$, and a value $\alpha \in (0, 1]$, ProMiSe identifies a set of significant sequential patterns with FWER bounded by α and frequency at least θ. ProMiSe starts by mining the set $FSP(D, \theta)$ of sequential patterns with frequency at least θ in D. It then uses the Monte Carlo procedure (Section II-D) to estimate the p-value $p_{\mathbf{s}}$ for each pattern $\mathbf{s}$ in $FSP(D, \theta)$, where the procedure RandomDataset(D) is used to sample a dataset uniformly at random among all datasets where each itemset appears the same number of times as in D and transactions have the same length as in D. A corrected significance threshold δ^* is then identified using the procedure CorrectedThreshold(α,θ). Such procedure employs the WY method (Section II-C) to compute δ^* and for each random dataset considers only patterns with frequency $\geq \theta$ since we are interested only in such patterns (see Section III-B for details). Finally, the output is given by all patterns in $FSP(D, \theta)$ with p-value at most δ^*. Note that ProMiSe crucially hinges on the ability of sampling random datasets with the desired properties described above.

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

Lemma 1: The output of ProMiSe has FWER bounded by α.

Algorithm 1: ProMiSe

Input: Sequential dataset $D = \{\tau_1, \tau_2, \ldots, \tau_{|D|}\}$;
minimum frequency threshold $\theta \in (0, 1]$;
FWER bound $\alpha \in (0, 1]$.
Output: Set of significant sequential patterns with FWER $\leq \alpha$.

```
F ← FSP(D, θ);
for i = 1 to T do
    D_i ← RandomDataset(D);
for s ∈ F do
    p_s ← 1/(T+1) (1 + Σ_{i=1}^{T} 𝟙[f_{D_i}(s) ≥ f_D(s)]);
δ* ← CorrectedThreshold(α,θ);
O ← {s ∈ F : p_s < δ*};
return O;
```

A. Efficiently Sampling Random Datasets

In this section we describe efficient methods to obtain a random dataset $\tilde{D}$ where:

- each itemset appears the same number of times as in D;
- each transaction has the same length as in D.

In particular, we consider two different strategies to obtain $\tilde{D}$. Both strategies start from D and perform random operations (*swaps* or *permutations*) at the level of itemsets. While both strategies preserve the two properties above, they focus also on preserving additional properties of D. Since the sequential patterns describe ordered sequences of events (i.e. itemsets), the idea is to preserve these events, represented by the itemsets, and to change only the order in which they occur. The two strategies are described in Section III-A1 and in Section III-A2.

1) Itemsets Swapping: We now describe a strategy similar to permutation swapping [21] previously proposed for significant itemsets mining [8], which cannot be directly used for sequential patterns since it does not take itemsets order into account and itemsets can appear more than once in a sequential transaction.

For each transaction $\tau_i \in D$, $i = 1, \ldots, |D|$, and each integer $j \in (0, \ldots, |\tau_i|]$, we use the pair (i, j) to represent the the j-th itemset in τ_i. Consider four positive integers i, j, k, ℓ such that $j \in (0, \ldots, |\tau_i|]$ and $\ell \in (0, \ldots, |\tau_k|)$.

In *itemset swapping* we swap the itemset (i, j) with the itemset (k, ℓ). As mentioned above, such swap preserve the length of each transaction. Note that the item-lengths of the transactions are not preserved, since the size of the two swapped itemsets may not be the same. The frequency of the swapped itemsets could change after a swap since the new transaction can contain itemsets that are super-sets of the ones swapped. For the same reason also the frequency of the items that compose the two itemsets may change as well.

Example. Consider the dataset in Section II-A. A possible swap is performed between itemset $\{1\}$ in position $(3, 1)$ and itemset $\{6, 7\}$ in position $(4, 2)$. The new dataset $\tilde{D}$ after the swap is the following: $\tau_1 =$

$\langle\{6,7\},\{7\},\{5\}\rangle; \tau_2 = \langle\{1,4\},\{3\},\{2\},\{1,2,5,6\}\rangle; \tau_3 = \langle\{6,7\},\{2\},\{6,7\},\{5\}\rangle; \tau_4 = \langle\{2\},\{1\}\rangle$. Note that the length of transactions τ_3 and τ_4 does not change after the swap, contrary to their item-length. The frequency of $\{1\}$ has remained the same, $f_D(\{1\}) = f_{\tilde{D}}(\{1\}) = 1/2$, while the frequency of $\{6,7\}$ has changed, $f_D(\{6,7\}) = 3/4 \neq 1/2 = f_{\tilde{D}}(\{6,7\})$.

Note that in a dataset D there are $m = \sum_{i=1}^{|D|} |\tau_i|$ total itemsets (not necessarily all distinct). Therefore, we can use the integer values ℓ with $1 \leq \ell \leq m$ to identify each itemset. Algorithm 2 shows how to generate a random dataset using a sequence of r itemsets swaps operations under this indexing of itemsets. Operation $swap(\tilde{D}, p_1, p_2)$ simply swaps the itemset of index p_1 with the itemset of index p_2 in $\tilde{D}$.

We now prove that the dataset produced in output by Algorithm 2 is a dataset taken uniformly at random among the set $\mathcal{D}$ of all datasets that satisfy the properties described in Section III-A, provided enough swap operations are performed. To analyze Algorithm 2 we use the Markov chains framework. Consider the Markov chain $M = \{S, T\}$, where S is the set space and T is the set of transitions. In our case, S is the set $\mathcal{D}$ of all datasets satisfying the two properties define in Section III-A, while T is defined in terms of *neighbors* of a dataset $D_j \in S$ where the neighbors of D_j are obtained by performing a swap operation on D_j. That is, the set T contains all pairs with datasets (D_j, D_k) such that it is possible to obtain D_k from D_j (or vice versa) with a single itemsets swap, with $D_j, D_k \in S$. For each dataset (state) $D_j \in S$, we define the *degree* of state D_j in Markov chain M as the number of possible swaps that can be performed in dataset D_j.

To prove that the output of Algorithm 2 produces in output a random dataset from $\mathcal{D}$ we prove that Markov chain M admits a unique stationary distribution π and that such distribution is uniform among all elements of $\mathcal{D}$.

Algorithm 2: RANDOMDATASET Generates a random dataset using the itemsets swaps operations.

Input: Sequential dataset $D = \{\tau_1, \tau_2, \ldots, \tau_{|D|}\}$.
Output: Random dataset $\tilde{D} = \{\tau'_1, \tau'_2, \ldots, \tau'_{|D|}\}$.
$\tilde{D} \leftarrow D$;
$m \leftarrow \sum_{i=1}^{|D|} |\tau_i|$;
for $i \leftarrow 1$ ***to*** r **do**
  $p_1 \leftarrow Random(1, m)$; $p_2 \leftarrow Random(1, m)$;
  $swap(\tilde{D}, p_1, p_2)$;
return $\tilde{D}$;

Theorem 1: The markov chain M admits as unique stationary distribution the uniform distribution.

Proof: Note that starting from any dataset D_j, it is possible to obtain any other dataset D_k with distinct single itemsets swaps operations, that is the Markov chain M is irreducible. The Markov chain M has a finite state space, it is irreducible, and it is aperiodic. A Markov chain with these properties is called ergodic. The Markov chain M is also reversible: an itemsets swap can be undone by a single (reversed) itemsets swap. From the theory of Markov chains [22], an ergodic Markov chain has a unique stationary distribution. From the reversibility properties, it follows that the probability of each state in such stationary distribution is proportional to the degree of the states. Therefore, in order to obtain a uniform distribution, all states of the Markov chain must have the same degree. Using Algorithm 2 to generate dataset $\tilde{D}$, all the states of the Markov chain have degree equals to m^2, with $m = \sum_{i=1}^{|D|} |\tau_i|$. This proves that the Markov chain M has a unique stationary distribution that is the uniform distribution. ■

In order to bound the number of swap operations that are required to converge to the stationary distribution we need to upper bound the *mixing time* of the Markov chain M. Upper bounding the mixing time is usually difficult; for example, for the mixing time of the commonly used swap randomization procedure [21], [23] has been the object of theoretical studies [21], but currently there are no conclusive results and only empirical analyses are available [8], [24].

We now prove an upper bound to the number of swap operations that are required for the Markov chain M to converge to the stationary distribution. In our proof we use the *path coupling* technique. In brief, given a Markov chain M, a coupling for M consists of two copies of the Markov chain M running simultaneously, where the two copies do not visit the states in the same order nor perform the same transition at the same time, but are defined on the same state space of M and have the same transition probabilities as M.

Theorem 2: The mixing time of Markov chain M is $\mathcal{O}(m^2 \log m)$, where $m = \sum_{i=1}^{|D|} |\tau_i|$.

Proof (sketch): We use path coupling to prove that after $\mathcal{O}(m^2 \log m)$ itemsets swaps, the Markov chain M converges to the stationary distribution. Let $D_\ell, D_h \in S$ be two datasets that differ only for the position of two itemsets. We say that D_ℓ, D_h are at distance $dist(D_\ell, D_h) = 2$. The idea is to start with a coupling for such pair of datasets, that differ in just two itemsets, and then extend the coupling over all pairs of datasets. Denoting with a and b the two positions where datasets D_ℓ and D_h differ, we define a coupling where the first Markov chain M_1 is M, while the second Markov chain M_2 is defined in terms of the transitions of M for a pair of datasets (D_ℓ, D'_ℓ). Let D_ℓ be the state of $M_1 = M$ at a given iteration, and let D_h be the state of M_2 at the same iteration. Let (D_ℓ, D'_ℓ) denote the transition performed by $M_1 = M$ (from state D_ℓ). In particular, let the itemsets swap performed by $M_1 = M$ be between the itemset in position p_1 and the itemset in position p_2 in dataset D_ℓ, (remember that p_1 and p_2 are sampled uniformly at random between all the positions). We then define the transitions (D_h, D'_h) for M_2 (from state D_h) as follows: i) if $p_1 = a$ and $p_2 = b$, M_2 swaps the itemset in position a with itself in the dataset D_h; ii) if $p_1 = b$ and $p_2 = a$, M_2 swaps the itemset in position b with itself in the dataset D_h; iii) if $p_1 = p_2 = a$ or $p_1 = p_2 = b$, M_2 swaps the itemset in position a with the itemset in position b in the dataset D_h; iv) otherwise M_2 swaps the itemset in position

p_1 with the itemset in position p_2 in the dataset D_h. Note that for both chains the probability of any given transition is still $1/m^2$. Most of the moves of such coupling maintain the distance $dist(D'_\ell, D'_h) = 2$. The only moves that result in $dist(D'_\ell, D'_h) = 0$ are the 4 moves described by i-iii, i.e. the moves that swap, in one of the two datasets, the two itemsets that are in different positions. Since each move occurs with the same probability $\frac{1}{m^2}$ and since $dist(D_\ell, D_h) = 2$ we have:

$$\mathbb{E}\left[dist(D'_\ell, D'_h)|D_\ell, D_h\right] = \left(1 - \frac{4}{m^2}\right) \cdot 2 + \frac{4}{m^2} \cdot 0 \leq (1 - \beta) \cdot dist(D_\ell, D_h),$$

with $\beta \geq \frac{4}{m^2}$. Thus by applying the path coupling theorem [22] we can extend the coupling to arbitrary pairs of states (D^0_ℓ, D^0_h) obtaining a bound on the mixing time of $\tau_{mix} = \mathcal{O}(\frac{1}{\beta} \log D)$, where D is the maximum distance between any two states. Since in our case we have $D = m$ and $\beta \geq \frac{4}{m^2}$ we obtain that the mixing time of M is bounded by $\tau_{mix} = \mathcal{O}(m^2 \log m)$. ∎

2) Random Permutations: We now introduce a different strategy to obtain a random dataset. This strategy produces a random dataset that still satisfies the properties on itemsets' presence and transactions' lengths defined in Section III-A, but it also forces itemsets to appear in the same transactions as in the original dataset D. In particular, this strategy ensures that only the *order* with which itemsets appear in each transaction is random, while everything else (e.g., itemsets frequencies, transactions in which each itemset appears, etc.) is fixed.

As a motivation, consider a dataset containing movies rated by some users. In such a dataset the items are the *ID*'s of the movies rated by the users. Each transaction contains the *ID*'s of the movies rated by a single user and movies rated in the same temporal interval, i.e. in the same day, are grouped in a single itemset. The transactions represent the temporal sequence of sets of movies rated in the same temporal interval. Since an user usually rates a movie only once, a movie is present at most once in each transaction. This feature of the data is not preserved by the random dataset generation strategy described in section III-A1.

Example. Consider again the dataset D in Section II-A. A random dataset $\tilde{D}$ generated from D using this approach is the following: $\tau_1 = \langle\{7\}, \{6, 7\}, \{5\}\rangle; \tau_2 = \langle\{3\}, \{1, 4\}, \{1, 2, 5, 6\}, \{2\}\rangle; \tau_3 = \langle\{5\}, \{1\}, \{6, 7\}, \{2\}\rangle; \tau_4 = \langle\{6, 7\}, \{2\}\rangle$. Note that each transaction of the new dataset has the same length, and also item-length, of the transactions of the dataset D. Also the frequency of the items and of the itemsets remains the same in the two datasets. Instead the frequency of the sequential patterns can change: $f_D(\langle\{1\}, \{2\}\rangle) = f_{\tilde{D}}(\langle\{1\}, \{2\}\rangle) = 1/2$ but $f_D(\langle\{6, 7\}, \{5\}\rangle) = 1/2 \neq 1/4 = f_{\tilde{D}}(\langle\{6, 7\}, \{5\}\rangle)$.

The *random permutation* strategy, described in Algorithm 3, produces a random dataset by permuting each transaction independently of all other events.[1]

[1] $\uplus$ in Algorithm 3 is the addition of an element to a multiset.

Algorithm 3: RANDOMDATASET Generates a random dataset by permuting each transaction independently.

Input: Sequential dataset $D = \{\tau_1, \tau_2, \ldots, \tau_{|D|}\}$.
Output: Random dataset $\tilde{D} = \{\tau'_1, \tau'_2, \ldots, \tau'_{|D|}\}$.
for $i \leftarrow 1$ ***to*** $|D|$ **do**
  $\tau'_i \leftarrow \texttt{Permute}(\tau_i)$;
  $\tilde{D} \leftarrow \tilde{D} \uplus \tau'_i$;
return $\tilde{D}$;

B. Parallel Implementation

Note that PROMISE, as all approaches based on permutation testing, is well-suited to parallelization. In particular, the generation of random datasets and the computation of p-values for patterns with frequency above θ in the dataset D can be easily parallelized: when k cores are used to compute the p-values using T permuted datasets in the Monte Carlo estimate (Section II-D), each core computes the p-values on T/k permuted datasets, and the results are then aggregated at the end.

Such parallel implementation is particularly advantageous for PROMISE, since the Monte Carlo estimate of patterns p-values is also required by the CORRECTEDTHRESHOLD(α,θ) procedure that computes, for each random dataset $D_i, i = 1, \ldots, P$, the minimum observed p-value in dataset D_i. That is, the p-values of patterns extracted from D_i are computed by analyzing the T permuted datasets with the parallel scheme described above.

Note that since we are interested only in patterns with frequency $\geq \theta$, the computation of the minimum p-values $p^{(1)}_{\min}, \ldots, p^{(P)}_{\min}$ for the WY method is restricted to patterns that have frequency $\geq \theta$ in the random datasets. That is, $p^{(i)}_{\min}$ is computed as the minimum p-value for patterns in $FSP(D_i, \theta)$ with frequency $\geq \theta$ in D_i. This corresponds to employ the WY method to estimate the probability of observing *any* pattern with frequency $\geq \theta$ and with p-value below the threshold δ under the null hypothesis, that is what we need in order to estimate FWER(δ) when we are interested only in patterns with frequency $\geq \theta$.[2] If $FSP(D_i, \theta) = \emptyset$, we then set $p^{(i)}_{\min} = \alpha$, corresponding to an uncorrected threshold. Note that any efficient implementation for mining frequent sequential patterns from a dataset can be used to obtain $FSP(D_i, \theta)$.

The final architecture of our implementation is presented in Figure 1, where the p-values of patterns in $FSP(D_i, \theta)$ for $i = 1, \ldots, P$ are computed using the parallel implementation of the Monte Carlo procedure, as described above.

IV. EXPERIMENTAL EVALUATION

We implemented and tested our algorithm PROMISE on real and synthetic data. Our experimental evaluation has three

[2] Note that in this way we are still allowing patterns $\mathbf{s}$ with frequency $f_D(\mathbf{s}) < \theta$ to appear with frequency higher than θ in a random dataset, thus properly accounting for such patterns in our multiple-hypothesis correction.

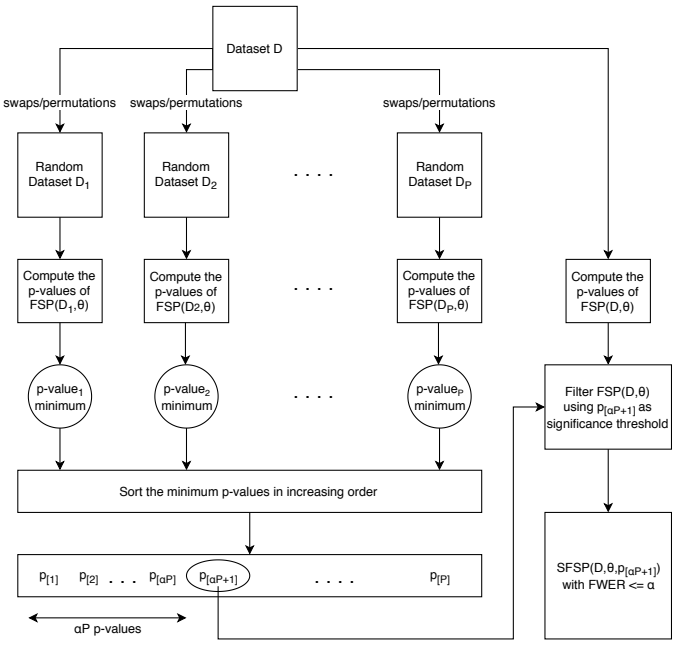

Fig. 1. Parallel implementation of PROMISE.

goals. First, to empirically estimate the number of swaps needed to reach the stationary distribution for the itemsets swaps strategy (see Section III-A1). Second, to evaluate the performance of our approach in terms of true and false discoveries, using random datasets. Third, to assess the performance of PROMISE on real datasets.

A. Implementation and Environment

The code used for the evaluation has been developed in Java and executed using version 1.8.0_201. Our experiments have been performed on a machine with 512 GB of RAM and 2 Intel(R) Xeon(R) CPU E5-2698 v3 @ 2.3GHz, with a total of 64 threads. To mine the sequential patterns we used the PrefixSpan [3] implementation provided by the SPMF Library [25]. The parallel algorithms have been implemented using the Apache Spark Java API version 2.2.0. We also used `fastutil`[3], a library that provides efficient data structures. The code developed for the tests and the implementation of PROMISE are available in our repository.[4]

B. Real data

We performed our experiments with 5 real sequential datasets:

- BIBLE[5]: A conversion of the Bible into a sequence dataset. A word is an item and a sentence corresponds to a sequence.
- BIKE[6]: Los Angeles Metro Bike Share Trip Data. An item is a bike-sharing station and a transaction is the sequence of bike-sharing stations in which a given bike was;
- FIFA[5]: A dataset of sequences of click-stream data from the website of FIFA World Cup 98. An item represents a web page;
- LEVIATHAN[5]: This dataset is a conversion of the novel Leviathan by Thomas Hobbes (1651) as a sequence database. A word is an item and a sentence corresponds to a sequence;
- SIGN[5]: a dataset of sign language utterance.

While BIBLE, FIFA, LEVIATHAN, and SIGN are public datasets that did not require any preprocessing, we preprocessed the BIKE dataset as described in Section IV-F. The characteristics of the datasets are reported in Table I.

TABLE I
DATASETS STATISTICS. FOR EACH DATASET THE TABLE REPORTS: THE NUMBER OF TRANSACTIONS $|D|$; THE NUMBER OF ITEMS $|\mathcal{I}|$; THE AVERAGE TRANSACTIONS ITEM-LENGTH; WHETHER ITEMS APPEAR MULTIPLE TIMES IN A TRANSACTION.

Dataset	Size $\|D\|$	$\|\mathcal{I}\|$	Avg. trans. item-length	Repeated items
BIBLE	36369	13905	21.6	Yes
BIKE	21078	67	7.28	Yes
FIFA	20450	2990	36.2	Yes
LEVIATHAN	5835	9025	33.8	Yes
SIGN	730	267	52.0	No

C. Convergence

In this section we empirically study the number of itemsets swaps needed to reach the stationary distribution, that is, to obtain a dataset drawn uniformly at random. Theorem 2 gives us an upper bound of the number of swaps to perform. In practice, the stationary distribution may be reached with a smaller number of swaps. To evaluate whether this is the case, we analyzed how the average relative frequency difference changed with the number of itemsets swaps.

Given the (real) dataset D, let the set $FSP(D,\theta)$ be the frequent sequential patterns extracted from D using θ as minimum frequency threshold, and $\tilde{D}_t$ be the (random) dataset obtained performing a series of t itemsets swaps from D. We define the average relative frequency difference $ARFD(\tilde{D}_t)$ for $\tilde{D}_t$ as $ARFD(\tilde{D}_t) = \frac{1}{|FSP(D,\theta)|}\sum_{\mathbf{s}\in FSP(D,\theta)}\frac{|f_D(\mathbf{s})-f_{\tilde{D}_t}(\mathbf{s})|}{f_D(\mathbf{s})}$.

If $ARFD(\tilde{D}_t)$ does not change much after $t > t^*$ swaps for some value t^*, then the distribution of the sequential patterns does not vary much in the datasets generated with more than t^* swaps, and the stationary distribution is reached with t^* swaps. We computed $ARFD(\tilde{D}_t)$ for $t = im$, where i is a positive integer and $m = \sum_{i=1}^{|D|}|\tau_i|$. For each real dataset we performed this computation five times (i.e., generating different random datasets $\tilde{D}_t$ starting from D) and then we compute the average value of $ARFD(\tilde{D}_{im})$. Figure 2 shows the value obtained with all the real datasets. For the dataset SIGN we showed all values of im up to $m^2 \log m$ (the value obtained with Theorem 2).

[3]http://fastutil.di.unimi.it
[4]Removed for anonymous submission. It will be available for camera ready.
[5]http://www.philippe-fournier-viger.com/spmf/index.php?link=datasets.php
[6]https://www.kaggle.com/cityofLA/los-angeles-metro-bike-share-trip-data

Note that just after $2m$ itemsets swaps, the $ARFD(\tilde{D}_m)$ of the generated dataset is very close to the one computed with the random dataset generated with $m^2 \log(m)$ itemsets swaps. For the other datasets we limited the number of swaps to be at most $100m$. For all datasets $2m$ swaps are

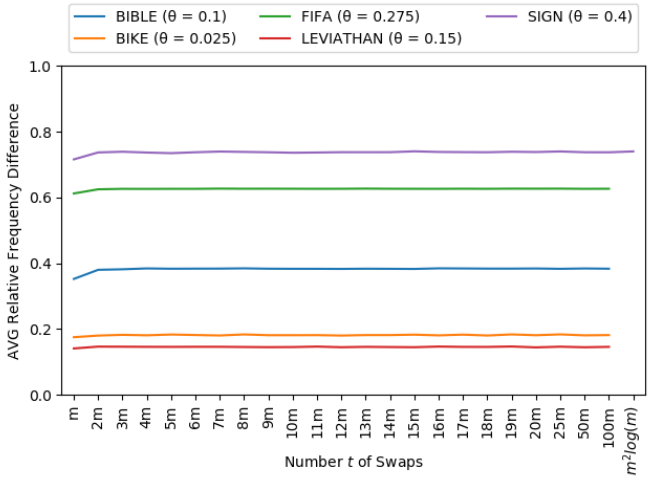

Fig. 2. Average relative difference $ARFD(\tilde{D}_t)$ between the frequencies of the frequent sequential patterns in D and their frequencies in random datasets generated with t itemsets swaps, for all datasets. Legend shows the values of θ used to compute the $ARFD$.

enough to convergence to the stationarity distribution. In all the experiments performed and reported below we consider a number of $2m$ itemsets swaps to generate a random dataset. Note that the $ARFD$ obtained with the same relative number of swaps in the five real datasets are very different. This is due to the different distributions of the sequential patterns in the datasets. In particular the datasets SIGN, that has values greater than the other datasets, does not contain repeated items in its transactions, and therefore a series of itemsets swaps changes its distribution more than on the other datasets.

D. Synthetic Data

To prove the correctness of our strategies, we ran PROMISE on random data, obtained with itemsets swaps or permutations, starting from BIKE. Since sequential patterns flagged as significant in a random datasets are false positives, we aimed to check if our algorithm returned some significant sequential patterns in some random datasets. We generated five random datasets for each generation strategy. In all cases PROMISE correctly did not return any significant sequential pattern.

To assess the ability of our method to identify significant patterns, we planted a sequential pattern of size 2 in the random datasets generated for the previous tests. For each transaction τ in the random datasets, we insert the pattern in τ with probability 0.6 independently of other events. We inserted the first itemset of the candidate sequential pattern in a random position inside the transaction, and the second itemset in a random position after the first itemset. We then run PROMISE with $\alpha = 0.05$, $P = 100$ and $T = 10048$: PROMISE correctly returned the planted pattern for all the datasets with both strategies.

E. Significant Sequential Patterns in Real Datasets

We used the SIGN dataset to evaluate the impact of the number P of permuted datasets for the WY method and of the number T of permuted datasets for the Monte Carlo estimate of the p-value. For all the tests we fixed the commonly used

TABLE II
RESULTS OF PROMISE FOR DATASET SIGN. FOR EACH TEST PERFORMED THE TABLE REPORTS: HOW RANDOM DATASETS ARE GENERATED (I.S. FOR ITEMSETS SWAPS, PERM. FOR PERMUTATIONS); THE NUMBER P OF RANDOM DATASETS USED FOR THE WY METHOD; THE NUMBER T OF RANDOM DATASETS FOR THE MONTE CARLO ESTIMATE OF p-VALUES; THE NUMBER FSP OF FREQUENT SEQUENTIAL PATTERNS FOUND IN THE DATASET; THE NUMBER $SFSP$ OF SIGNIFICANT FSP RETURNED BY PROMISE.

Dataset	Strategy	P	T	FSP	SFSP
SIGN ($\theta = 0.4$)	I.S.	1000	1024	518	518
		100	10048		518
		1000	10048		518
		100	100032		518
	Perm.	1000	1024	518	/
		100	10048		457
		1000	10048		457
		100	100032		457

value $\alpha = 0.05$ as FWER threshold. We tested multiple combinations of the parameters P and T. We tested values of T that are multiples of 64 (the number of cores of our machine). Table II reports the tests we performed and their results. The tests with the itemsets swaps flagged as significant all the frequent sequential patterns considered, with all the combinations of P and T: this is due to the absence of repeated items in the transactions of SIGN that is therefore extremely different from a random dataset (see Section IV-C). With permutations, no significant sequential patterns had been found with $T = 1024$, since the Monte Carlo estimates of the p-values are too coarse (the smallest p-value that can be estimated is $1/1025$). For all the other combinations, the number of significant sequential patterns returned by the algorithm remains the same. We therefore fixed the values $P = 100$ and $T = 10048$ for all the remaining analyses.

TABLE III
RESULTS OF PROMISE FOR FOUR REAL DATASETS. SEE TABLE II FOR THE MEANING OF THE REPORTED VALUES.

Dataset	Strategy	P	T	FSP	SFSP
BIBLE ($\theta = 0.1$)	I.S.	100	10048	174	120
	Perm.				121
BIKE ($\theta = 0.025$)	I.S.	100	10048	163	37
	Perm.				31
FIFA ($\theta = 0.275$)	I.S.	100	10048	182	181
	Perm.				142
LEVIATHAN ($\theta = 0.15$)	I.S.	100	10048	225	87
	Perm.				100

Tables III reports the results obtained for other real datasets. For all the datasets and for both generation strategies, PROMISE returned some significant sequential patterns. In

addiction the sequential patterns flagged as significant are always less than the frequent sequential patterns, confirming that frequency by itself is not enough to provide statistical significance. For the FIFA dataset, only one sequential pattern is flagged as non significant using itemsets swaps. This is due to the fact that such dataset contains very few transactions with repeated itemsets, therefore, similarly to what we observed for SIGN, the itemsets swaps greatly change the distribution of patterns' frequencies. Note that for some datasets, the number of significant sequential patterns found with the permutations are greater than the one found with the itemsets swaps, though the permutations can not detect sequential patterns of length one or sequential patterns composed by a single itemset repeated multiple times. Such results depend on the distribution of the sequential patterns in the real datasets. This highlights that our two strategies for datasets generation provide complementary assessments of patterns' significance.

Table IV reports the execution times required to obtain a single random dataset, for both strategies. For the itemsets swaps we considered $2m$ swaps. Table IV also reports the time required for mining a random dataset in order to compute the supports of the frequent sequential patterns. For each dataset the minimum frequency threshold θ is shown in Table III. Table IV shows that our strategies to generate random datasets is very efficient, with all generation times being lower than 1 second. As expected, the generation of the datasets using the permutations approach is faster than the approach with itemsets swaps, since it only requires to permute the itemsets in each transaction. For both strategies the time to generate a random dataset is usually smaller or at most comparable to the time for mining a (random) dataset.

TABLE IV
EXECUTION TIME FOR THE GENERATION OF A RANDOM DATASET AND EXECUTION TIME TO MINE THE RANDOM DATASETS WITH BOTH GENERATION STRATEGIES (I.S. FOR ITEMSETS SWAPS, PERM. FOR PERMUTATIONS).

Dataset	Execution Time (ms)			
	Dataset Generation		Mining	
	I.S.	Perm.	I.S.	Perm.
BIBLE	624.84	22.35	525.23	445.01
BIKE	73.32	4.13	127.03	114.79
FIFA	595.26	18.42	409.29	378.44
LEVIATHAN	96.82	7.71	164.38	159.55
SIGN	11.85	1.73	40.81	32.45

Since the number of items in BIKE is relative small, we also tested the use of the Bonferroni correction and we compared the results with the ones obtained with PROMISE. Note that BIKE is the only dataset that allows this type of tests, since for the other datasets the Bonferroni correction results in a very small corrected threshold and thus requires an extremely large number of random datasets to identify a pattern as significant. For the test with the itemsets swaps we considered sequential patterns of sizes 1 and 2. Since the itemsets of BIKE are composed by single items, the number of sequential patterns of size 1 and 2 are respectively $|\mathcal{I}|$ and $|\mathcal{I}|^2$, thus we used $\alpha/(|\mathcal{I}| + |\mathcal{I}|^2)$ as corrected significance threshold. For the test with the permutations, we considered only the sequential patterns of size 2, since the ones of size 1 can not be flagged as significant. For both tests, we generated $T = 512 \cdot 10^3$ datasets for the Monte Carlo estimate of the p-values. The minimum frequency threshold θ is the one showed in Table III. Table V shows the results obtained. For both strategies, PROMISE, that use the WY method, returns a number of significant sequential patterns larger than the ones returned using the Bonferroni correction. Note that all the significant sequential patterns returned using the Bonferroni correction are returned also by PROMISE (for either strategies, itemsets swaps and permutations, to generate random datasets).

TABLE V
SIGNIFICANT SEQUENTIAL PATTERN MINING FOUND WITH BONFERRONI CORRECTION IN BIKE. FOR EACH TEST PERFORMED THE TABLE REPORTS: THE SIZE L OF THE FREQUENT SEQUENTIAL PATTERNS CONSIDERED; THE TOTAL NUMBER $\#SP$ OF SEQUENTIAL PATTERNS OF THAT SIZE CONSIDERED; SEE TABLE II FOR THE MEANING OF THE OTHER REPORTED VALUES.

L	Strategy	FSP	# SP	$SFSP$	T
1 and 2	I.S.	163	4556	36	$512 \cdot 10^3$
2	Perm.	163	4489	28	$512 \cdot 10^3$

F. Analysis of BIKE's Results

The BIKE dataset is obtained from public data available online and therefore allows us to analyze the results more in detail. We obtained the BIKE dataset by downloading the Los Angeles Metro Bike Share Trip Data and performing the following preprocessing steps in order to create a sequential dataset. The Los Angeles Metro Bike Share Trip Data contains a series of trips performed in Los Angeles using the bike sharing service. For each trips the following information are available, among others: the starting station, the ending station, an unique identifier of the bike, the starting time, the ending time. We collected all the trips made with each bike and we sort them using the temporal information available. In particular for each bike we collect the ordered sequence of station where the bike was. The items in our dataset are the ID of the stations. Each transaction represents the ordered list of stations in which a certain bike was. The first station in each transaction is the starting station of the first trip available for that bike. The second is the ending station of that trip, which becomes the starting station for the following trip and so on. If for some reason one series of stations contained gaps, i.e. the end station of a trip does not correspond to the starting station of the next trip, we split the sequence where the gap happens creating two transactions. We extracted the significant sequential patterns from the BIKE dataset with PROMISE using the parameters showed in Table III. Figure 3 shows the 4 most significant (i.e., with smallest p-value) sequential patterns with highest frequency found with itemsets swaps, and the 4 most significant ones with highest frequency found with the permutations (without considering the 4 patterns from itemsets swap). The two patterns found only with itemsets swaps can not be detected using permutations, since they are composed

by two equal itemsets and such patterns have always the same frequency in random datasets generated with permutations. Our algorithm flagged as significant the sequential pattern starting from Union Station West Portal and ending in the same bike station. Such bike station is located near Union Station, the main train station of Los Angeles. Another significant sequential pattern, analogous to this one, starts from 7th & Flower and ends again in the same station. Such station is located near the Metro Center station in the financial district. We investigated if in the transactions the two itemsets that composed these sequential patterns are consecutive, indicating that they correspond to single trips or if they are a combination of multiple trips. We considered that the two itemsets of a sequential pattern are consecutive in a transaction if there was at least one instance of such pattern with itemsets in consecutive positions in the transaction. For both the sequential patterns that start and end in the same station, the percentage of single trips is about 25%. The only significant sequential pattern that has a lower percentage of single trips is the one that starts from Main & First and that ends in 7th & Flower. This is probably due to the large distance between the two stations. Instead the pattern with higher probability of single trips is the one that starts from Main & First and that ends in Union Station West Portal. This pattern probably catches the flow of people that ride to Union Station.

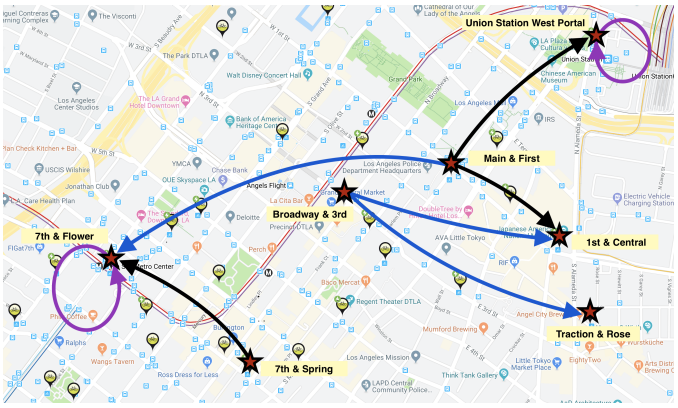

Fig. 3. Map of the Los Angeles Metro Bike stations with some significant sequential patterns found with PROMISE. The red stars are the bike stations involved, with respective names. Each arrow is a significant sequential pattern. Black arrows are significant sequential patterns found with both dataset generation strategies. Blue arrows are found only with permutations while purple arrows only with itemsets swaps.

CONCLUSION

We presented PROMISE, an efficient algorithm to extract sequential patterns with a frequency that is significantly higher than expected in random datasets where both the number of times each itemset appears in the dataset and the length of the transactions are as observed in the data. We proposed two efficient strategies to sample random datasets from such null model and employed them within a WY scheme to properly control the FWER. Our extensive experimental evaluation shows that PROMISE efficiently identifies significant sequential patterns in real datasets. There are several directions for future research, including using our randomization techniques to identify significant patterns while controlling the False Discovery Rate (FDR) and devising efficient strategies to identify the k most significant sequential patterns.

ACKNOWLEDGMENT

This work is supported, in part, by the U. of Padova grants *SID2017* and *STARS: Algorithms for Inferential Data Mining*.

REFERENCES

[1] R. Agrawal, R. Srikant, *et al.*, "Mining sequential patterns," *ICDE* 1995.
[2] J. Han, J. Pei, B. Mortazavi-Asl, Q. Chen, U. Dayal, and M.-C. Hsu, "Freespan: frequent pattern-projected sequential pattern mining," *ACM SIGKDD* 2000.
[3] J. Pei, J. Han, B. Mortazavi-Asl, J. Wang, H. Pinto, Q. Chen, U. Dayal, and M.-C. Hsu, "Mining sequential patterns by pattern-growth: The prefixspan approach," *IEEE TKDE* 2004.
[4] C. Raïssi, T. Calders, and P. Poncelet, "Mining conjunctive sequential patterns," *Data Mining and Knowl. Discovery* 2008.
[5] X. Yan, J. Han, and R. Afshar, "Clospan: Mining: Closed sequential patterns in large datasets," *ICDM* 2003.
[6] A. Kirsch, M. Mitzenmacher, A. Pietracaprina, G. Pucci, E. Upfal, and F. Vandin, "An efficient rigorous approach for identifying statistically significant frequent itemsets," *JACM* 2012.
[7] G. I. Webb, "Discovering significant patterns," *Machine learning* 2007.
[8] A. Gionis, H. Mannila, T. Mielikäinen, and P. Tsaparas, "Assessing data mining results via swap randomization," *ACM TKDD* 2007.
[9] W. Hämäläinen and M. Nykänen, "Efficient discovery of statistically significant association rules," *ICDM* 2008.
[10] R. Gwadera and F. Crestani, "Ranking sequential patterns with respect to significance," *PAKDD* 2010.
[11] C. Low-Kam, C. Raïssi, M. Kaytoue, and J. Pei, "Mining statistically significant sequential patterns," *ICDM* 2013.
[12] J. Han, H. Cheng, D. Xin, and X. Yan, "Frequent pattern mining: current status and future directions," *Data mining and knowl. discovery* 2007.
[13] A. Terada, M. Okada-Hatakeyama, K. Tsuda, and J. Sese, "Statistical significance of combinatorial regulations," *PNAS* 2013.
[14] S.-i. Minato, T. Uno, K. Tsuda, A. Terada, and J. Sese, "A fast method of statistical assessment for combinatorial hypotheses based on frequent itemset enumeration," *ECML-PKDD* 2014.
[15] F. Llinares-López, M. Sugiyama, L. Papaxanthos, and K. Borgwardt, "Fast and memory-efficient significant pattern mining via permutation testing," *ACM SIGKDD* 2015.
[16] L. Pellegrina and F. Vandin, "Efficient mining of the most significant patterns with permutation testing," *ACM SIGKDD* 2018.
[17] M. Boley, T. Horváth, and S. Wrobel, "Efficient discovery of interesting patterns based on strong closedness," *Statistical Analysis and Data Mining: The ASA Data Science Journal* 2009.
[18] N. Tatti and M. Mampaey, "Using background knowledge to rank itemsets," *Data Mining and Knowledge Discovery* 2010.
[19] M. Mampaey, J. Vreeken, and N. Tatti, "Summarizing data succinctly with the most informative itemsets," *ACM TKDD* 2012.
[20] P. H. Westfall and S. S. Young, "Resampling-based multiple testing: Examples and methods for p-value adjustment," *Wiley Series in Probability and Statistics* 1993.
[21] G. W. Cobb and Y.-P. Chen, "An application of markov chain monte carlo to community ecology," *The American Math. Monthly* 2003.
[22] M. Mitzenmacher and E. Upfal, *Probability and computing*. Cambridge university press, 2017.
[23] A. Gobbi, F. Iorio, K. J. Dawson, D. C. Wedge, D. Tamborero, L. B. Alexandrov, N. Lopez-Bigas, M. J. Garnett, G. Jurman, and J. Saez-Rodriguez, "Fast randomization of large genomic datasets while preserving alteration counts," *Bioinformatics* 2014.
[24] R. Milo, N. Kashtan, S. Itzkovitz, M. E. Newman, and U. Alon, "On the uniform generation of random graphs with prescribed degree sequences," *arXiv preprint cond-mat/0312028* 2003.
[25] P. Fournier Viger, C.-W. Lin, A. Gomariz, T. Gueniche, A. Soltani, Z.-H. Deng, and H. Thanh Lam, "The spmf open-source data mining library version 2," 2016.