Occupancy-Based Frequent Pattern Mining*

LEI ZHANG, Anhui University
PING LUO, University of Chinese Academy of Sciences
LINPENG TANG, Princeton University
ENHONG CHEN and QI LIU, University of Science and Technology of China
MIN WANG, Google Research
HUI XIONG, Rutgers, the State University of New Jersey

Frequent pattern mining is an important data mining problem with many broad applications. Most studies in this field use *support* (frequency) to measure the *popularity* of a pattern, namely the fraction of transactions or sequences that include the pattern in a data set. In this study, we introduce a new interesting measure, namely occupancy, to measure the *completeness* of a pattern in its supporting transactions or sequences. This is motivated by some real-world pattern recommendation applications in which an interesting pattern should not only be frequent, but also occupies a large portion of its supporting transactions or sequences. With the definition of occupancy we call a pattern *dominant* if its occupancy value is above a user-specified threshold. Then, our task is to identify the *qualified patterns* which are both dominant and frequent. Also, we formulate the problem of *mining top-k qualified patterns*, that is, finding *k* qualified patterns with maximum values on a user-defined function of support and occupancy, for example, weighted sum of support and occupancy. The challenge to these tasks is that the value of occupancy does not change monotonically when more items are appended to a given pattern. Therefore, we propose a general algorithm called DOFRA (DOminant and FRequent pattern mining Algorithm) for mining these qualified patterns, which explores the upper bound properties on occupancy to drastically reduce the search process. Finally, we show the effectiveness of DOFRA in two real-world applications and also demonstrate the efficiency of DOFRA on several real and large synthetic datasets.

*This article is a substantially extended and revised version of Tang et al. [2012], which appeared in Proceedings of ACM 2012 International Conference on Information and Knowledge Management (CIKM'2012) as the Best Student Paper.

This work was supported in part by the Academic and Technology Leader Imported Project of Anhui University (No. J10117700050) and the Natural Science Foundation of China (Grant No. 61502001). This work was also partially supported by grants from the National Science Foundation for Distinguished Young Scholars of China (Grant No. 61325010), the Natural Science Foundation of China (Grant No. 61403358 and 61473274) and National 863 Program (No. 2014AA015105). Qi Liu gratefully acknowledges the support of the Youth Innovation Promotion Association, CAS.

Authors' addresses: L. Zhang, Key Laboratory of Intelligent Computing & Signal Processing, Ministry of Education, Department of Computer Science and Technology, Anhui University, No. 3 Feixi Road, Hefei, Anhui Province 230039, China; emails: zl@ahu.edu.cn; P. Luo (corresponding author), University of Chinese Academy of Sciences, Beijing 10010, China; emails: luop@ict.ac.cn; L. Tang, Computer Science Department, Princeton University, Princeton, NJ08544, USA; email: chnttlp@gmail.com; E. Chen (corresponding author) and Q. Liu, School of Computer Science and Technology, University of Science and Technology of China, Hefei, Anhui 230026, China; emails: {cheneh, qiliuql}@ustc.edu.cn; M. Wang, Google Research, Parkway Mountain View, CA 94043, USA; email: minwang@google.com; H. Xiong, Management Science and Information Systems Department Rutgers, the State University of New Jersey, Newark, NJ07102, USA; email: hxiong@rutgers.edu.

DOI: http://dx.doi.org/10.1145/2753765

Categories and Subject Descriptors: H.2.8 [**Database Management**]: Database Applications—*Data mining*

General Terms: Methods, Algorithms

Additional Key Words and Phrases: Occupancy, dominant pattern, bound estimation

ACM Reference Format:
Lei Zhang, Ping Luo, Linpeng Tang, Enhong Chen, Qi Liu, Min Wang, and Hui Xiong. 2015. Occupancy-based frequent pattern mining. ACM Trans. Knowl. Discov. Data 10, 2, Article 14 (October 2015), 33 pages.
DOI: http://dx.doi.org/10.1145/2753765

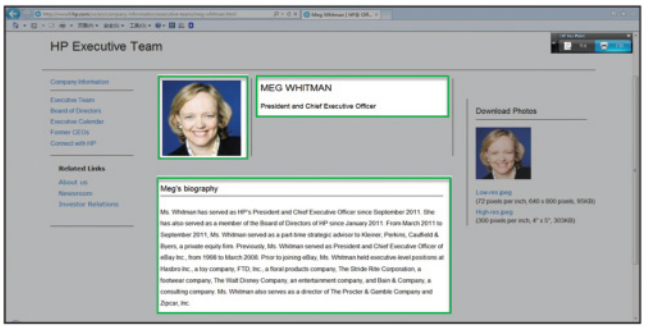
1. INTRODUCTION

Frequent pattern mining [Agrawal et al. 1993] is an important data mining problem in many data mining tasks, such as association-rule-based classification [She et al. 2003] and clustering [Aggarwal et al. 2007]. Most of studies in this field use *support* (frequency) to measure a pattern's *popularity*, namely the fraction of transactions orsequences that include the pattern in a database. Here, the frequent patterns can be itemsets or subsequences that appear in a data set with frequency no less than a user-specified threshold [Han et al. 2007]. In this article, we introduce a new concept *occupancy* to measure a pattern's *completeness*, namely the degree to which the pattern occupies its supporting transactions or sequences. This new measure is motivated by some real-world applications of pattern recommendation which not only require that an interesting pattern should be frequent, but also prefer the pattern which occupies a large portion of its supporting transactions or sequences. Next, we will give two of such applications on a transaction database and a sequence database, respectively.

The first motivating application on a transaction database is the print-area recommendation for Web pages [Tang et al. 2012]. Users may have the experience that the printout generated by a Web browser's print function is far from satisfactory, since it usually contains many irrelevant contents (e.g., advertisements, navigation menu, and related links). To tackle this problem, HP designs a tool named *Smart Print*[1], which provides a user-friendly interface so that a user can easily select her interested print areas for a Web page. Figure 1 gives an example for printing the biography of HP CEO in Smart Print. In this Web page, the user selected three print-areas which are highlighted in white rectangles. A lot of such selections for Web pages are stored in print logs with user content. If we view each content clip (a selected content area) as an item, then all the selected clips by a user in a given Web page form a transaction of items and the print log data from all users form a transaction database. Based on this transaction database, our task is to recommend an itemset (i.e., a set of content clips) in a given Web page to users. In this task, the recommended itemset should not only occur frequently to reflect the interests of most users, but also occupy a large portion of its supporting transactions to ensure the completeness of the itemset so that the user would not feel that the recommendation is missing too much relevant content.

Another motivating application on a sequence database is the travel landscapes recommendation [Liu et al. 2011]. Assume that we have the travel package consumed logs from the tourists in a travel company. Here, a travel package means the comprehensive services provided by a travel company for the tourists based on their travel preference, which usually consists of many landscapes by a spatial or temporal order. Figure 2 gives an example document for a travel package named "Niagara Falls Discovery" from the STA Travel[2]. In this document, the words in red are the landscapes appearing in a sequential order. If we view each landscape as an item, the package purchased by a tourist form a sequence of items. Then, the purchase log data from all tourists

[1]A Web browser plug-in, www.hp.com/go/smartprint.
[2]http://www.statravel.com/.


HP Executive Team

Company Information

Executive Team
Board of Directors
Executive Calendar
Former CEOs
Connect with HP

Related Links
About us
Newsroom
Investor Relations

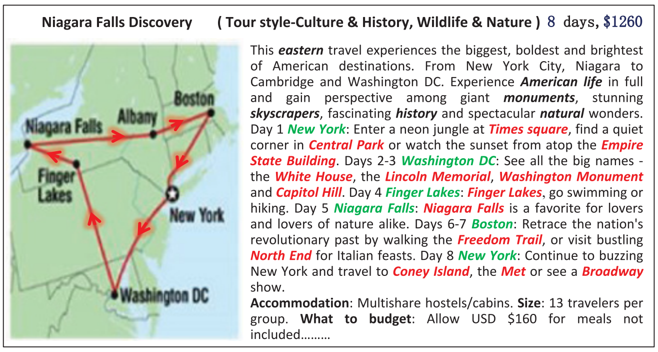
MEG WHITMAN
President and Chief Executive Officer

Meg's biography

Ms. Whitman has served as HP's President and Chief Executive Officer since September 2011. She has also served as a member of the Board of Directors of HP since January 2011. From March 2011 to September 2011, Ms. Whitman served as a part-time strategic advisor to Kleiner, Perkins, Caufield & Byers, a private equity firm. Previously, Ms. Whitman served as President and Chief Executive Officer of eBay Inc., from 1998 to March 2008. Prior to joining eBay, Ms. Whitman held executive-level positions at Hasbro Inc., a toy company, FTD, Inc., a floral products company, The Stride Rite Corporation, a footwear company, The Walt Disney Company, an entertainment company, and Bain & Company, a consulting company. Ms. Whitman also serves as a director of The Procter & Gamble Company and Zipcar, Inc.

Download Photos
Low-res jpeg
(72 pixels per inch, 640 x 800 pixels, 95KB)
High-res jpeg
(300 pixels per inch, 4" x 5", 303KB)

Fig. 1. An example for printing in Smart Print.


Niagara Falls
Albany
Boston
Finger Lakes
New York
Washington DC

Niagara Falls Discovery **(Tour style-Culture & History, Wildlife & Nature)** 8 days, $1260

This ***eastern*** travel experiences the biggest, boldest and brightest of American destinations. From New York City, Niagara to Cambridge and Washington DC. Experience ***American life*** in full and gain perspective among giant ***monuments***, stunning ***skyscrapers***, fascinating ***history*** and spectacular ***natural*** wonders. Day 1 ***New York***: Enter a neon jungle at ***Times square***, find a quiet corner in ***Central Park*** or watch the sunset from atop the ***Empire State Building***. Days 2-3 ***Washington DC***: See all the big names - the ***White House***, the ***Lincoln Memorial***, ***Washington Monument*** and ***Capitol Hill***. Day 4 ***Finger Lakes***: ***Finger Lakes***, go swimming or hiking. Day 5 ***Niagara Falls***: ***Niagara Falls*** is a favorite for lovers and lovers of nature alike. Days 6-7 ***Boston***: Retrace the nation's revolutionary past by walking the ***Freedom Trail***, or visit bustling ***North End*** for Italian feasts. Day 8 ***New York***: Continue to buzzing New York and travel to ***Coney Island***, the ***Met*** or see a ***Broadway*** show.
Accommodation: Multishare hostels/cabins. **Size**: 13 travelers per group. **What to budget**: Allow USD $160 for meals not included………

Fig. 2. An example of a travel package document.

form a sequence database. Our task is to recommend a sequence of landscapes to a given tourist based on this sequence database. Intuitively, the recommended sequence should occur frequently to reflect the preference of most tourists. More importantly, the recommended sequence should be as complete as possible so that the user would not miss too much interesting landscapes.

The commonality of the previous two applications is that the items in each transaction or sequence *work as a whole for a task*. To be specific, the task of print-area recommendation is to recommend a set of content clips in a given Web page to users, while that of travel-landscape recommendation is to recommend a sequence of landscapes to a given tourist. In these applications, the recommendation for the tasks should be both precise and complete. Intuitively, the support of a recommended pattern correlates to the recommendation precision while its occupancy is related to the recommendation recall. Thus, occupancy become another pattern interestingness measure which is an indispensable complement to support.

With the definition of occupancy, we call a pattern *dominant* if its occupancy is above a user-specified minimal threshold. Then, our task is to identify the qualified patterns which are both dominant and frequent. Also, we formulate the problem of *mining top-k qualified patterns*, that is, finding *k* qualified patterns with maximum values on a user-defined function of support and occupancy (e.g., weighted sum of support and occupancy). The challenge to these tasks is that the property of occupancy is neither monotone nor anti-monotone. In other words, the value of occupancy does not increase or decrease monotonically when more items are appended to a given pattern.

To address this challenge, in our preliminary work [Tang et al. 2012], we proposed an efficient algorithm, named DOFIA (DOminant and Frequent Itemset mining

Algorithm), which incorporates occupancy into frequent itemset mining for high quality itemset recommendation. DOFIA explores the occupancy upper bound properties to drastically reduce the search process, thus can increase the efficiency of mining. However, DOFIA can only be applied to transaction databases for itemset mining.

In this article, we extend the concept of occupancy into sequential pattern mining and propose an efficient algorithm called DOFRA for high quality sequential pattern recommendation. For the scenarios of sequential pattern mining, we propose two upper bounds on occupancy for any given itemset X and its supersets (in the traversal tree). The first bound is efficient and the second one is the tightest with certain input constraint. Also, we show the technique that achieves the balance between the two upper bounds to improve the overall performance. Moreover, we show that these techniques can be also applied to *weighted occupancy*, in which we consider the weights of items in calculating the pattern occupancy.

Specifically, we propose two occupancy definitions, namely *harmonic occupancy* and *arithmetic occupancy*, in which different average functions, namely *harmonic average* and *arithmetic average*, are used respectively. In our preliminary work [Tang et al. 2012], we focused on harmonic occupancy. In case that in some applications the arithmetic average is required, we focus on arithmetic occupancy in this study for technical complement to our previous work [Tang et al. 2012]. In addition, the link between occupancy and closed patterns has been investigated. To be specific, the top qualified pattern must be closed one, and thus DOFRA can be enhanced by exploiting the properties of closed patterns to further prune the search space. Finally, we show the effectiveness of DOFRA in the two mentioned applications, namely the print-area recommendation and the travel-landscape recommendation. And, we also demonstrate the efficiency of DOFRA on several real and large synthetic datasets.

Overview: The remainder of the article is organized as follows. We first give the problem formulation in Section 2 and then give the overview of the proposed algorithm DOFRA in Section 3. The details of DOFRA, especially on how to calculate the upper bounds on occupancy and quality, are discussed in Section 4. We report the empirical results to show the effectiveness and efficiency of DOFRA in Section 5 and 6, respectively. We present the related works in Section 7 and conclude the article in Section 8.

2. PRELIMINARIES AND PROBLEM FORMULATION

In this section, we first give some preliminaries in pattern mining on a sequence database and propose the definitions of occupancy. Then, we formulate the qualified pattern mining problem which considers both support and occupancy.

2.1. Definitions and Notations

Let $\mathcal{I}$ be the complete set of distinct items. A *sequence database* is a set of sequences, in which each *sequence* is an ordered list of events which is denoted as $e_1e_2\ldots e_m$. If each event in a given sequence is an *item*, we call it *single-itemset* sequence. Otherwise, we call it *multiple-itemset* sequence in which each event is an *itemset*. In this article, we only focus on *single-itemset* sequences since it represents one of the most important and popular type of sequences, such as protein sequences, DNA strings, Web click streams and the travel-package sequences introduced in this article.

We call a subsequence $S_a = a_1a_2\ldots a_m$ *contains* another subsequence $S_b = b_1b_2\ldots b_n$ if there exist integers $1 \leq i_1 < i_2 < \cdots < i_n \leq m$ such that $a_{i_1} = b_1, a_{i_2} = b_2, \ldots, a_{i_n} = b_n$ (Note that an item can occur multiple times in different events of a sequence). For example, suppose $S_1 = abcdb$, $S_2 = acd$ and $S_3 = adca$, then S_1 contains S_2, but S_1 does not contain S_3.

Without loss of generality, we use the pattern P to represent a subsequence and the database $\mathcal{D}$ to represent a sequence database. The number of items in P, denoted by

Table I. An Example of Sequence Database

S_{id}	Sequence	Length
S_1	a b c d e g	6
S_2	a b c f	4
S_3	a b c	3
S_4	a b c	3
S_5	a b e h i	5

$|P|$, is called the length of P and a pattern with a length l is called a *l-pattern*. The sequences in $\mathcal{D}$ that contain a pattern P are the *supporting sequences* of P, denoted as $\mathcal{D}_P$. The *frequency* of a pattern P in $\mathcal{D}$ is defined as $freq(P) = |\mathcal{D}_P|$. The *support* of a pattern P is the percentage of sequences in $\mathcal{D}$ that containing P which is defined as $\sigma(P) = freq(P)/|\mathcal{D}|$. For a given minimum support threshold α $(0 < \alpha \leq 1)$, a pattern P is said to be *frequent* if $\sigma(P) \geq \alpha$.

Table I gives an example of sequence database $\mathcal{D}$. This database has nine unique items $(a, b, c, d, e, f, g, h, i)$, five input sequences $(|\mathcal{D}| = 5)$. Suppose $P = abc$, there are four sequences (S_1, S_2, S_3, S_4) containing this pattern, thus $freq(P) = 4$ and $\sigma(P) = \frac{4}{5}$. P is frequent if $\alpha = 0.5$.

In the motivating applications in Section 1, the pattern we intend to find should occupy a large portion of its supporting sequences so that the recommended pattern contains most of interesting items without missing anything. The occupancy can be calculated as follows. For a pattern P, we can identify all its supporting sequences $\mathcal{D}_P$. For each sequence $S \in \mathcal{D}_P$, we calculate the ratio of $\frac{|P|}{|S|}$ to measure the completeness of P in S. For example, the ratio of $P = abc$ in S_1 shown in Table I is 3/6. We then aggregate these ratios to compute a single value of occupancy for P in its supporting sequences.

Definition 2.1 (*Occupancy*). The *occupancy* of a pattern P is defined as

$$\phi(P) = AVG\left(\left\{\frac{|P|}{|S|} : S \in \mathcal{D}_P\right\}\right),$$

where $AVG()$ is an average function of all the values in the set.

Different average functions, *arithmetic average* and *harmonic average* [Haff 1979] can be used here. Then, we have the following two occupancy definitions.

Definition 2.2 (*Harmonic occupancy*). The *harmonic occupancy* is defined as

$$\phi_H(P) = AVG_H\left(\left\{\frac{|P|}{|S|} : S \in \mathcal{D}_P\right\}\right) = \frac{|\mathcal{D}_P||P|}{\sum_{S\in\mathcal{D}_P}|S|},$$

where $AVG_H(X) = \frac{|X|}{\sum_{x\in X}\frac{1}{x}}$ is the harmonic average of a set of numbers in X.

Definition 2.3 (*Arithmetic occupancy*). The *arithmetic occupancy* is defined as

$$\phi_A(P) = AVG_A\left(\left\{\frac{|P|}{|S|} : S \in \mathcal{D}_P\right\}\right) = \frac{1}{|\mathcal{D}_P|}\sum_{S\in\mathcal{D}_P}\frac{|P|}{|S|},$$

where $AVG_A(X) = \frac{\sum_{x\in X} x}{|X|}$ is the arithmetic average of a set of numbers in X.

Let us take the database in Table I as an example. We calculate the occupancy of the following three patterns $P_1 = ab$, $P_2 = abc$, and $P_3 = abcd$. The supporting sequences of P_1, P_2, and P_3 are $\{S_1, S_2, S_3, S_4, S_5\}$, $\{S_1, S_2, S_3, S_4\}$, and $\{S_1\}$ respectively. Then,

$$\begin{aligned}
\phi_H(P_1) &= \tfrac{5\times 2}{6+4+3+3+5} \approx 0.48\\
\phi_H(P_2) &= \tfrac{4\times 3}{6+4+3+3} = 0.75\\
\phi_H(P_3) &= \tfrac{4}{6} \approx 0.67.\\
\phi_A(P_1) &= \tfrac{\frac{2}{6}+\frac{2}{4}+\frac{2}{3}+\frac{2}{3}+\frac{2}{5}}{5} \approx 0.51\\
\phi_A(P_2) &= \tfrac{\frac{3}{6}+\frac{3}{4}+\frac{3}{3}+\frac{3}{3}}{4} \approx 0.81\\
\phi_A(P_3) &= \tfrac{4}{6} \approx 0.67
\end{aligned}$$

It is clear that the occupancy of a pattern P is the average ratio of the occurrences of the items in P to the number of the items in its supporting sequence. The high value of the occupancy indicates that besides the items in P there are only a small number of items left inside the supporting sequence of P. Also, it is easy to check that the value of occupancy does not increase (see $\phi(P_2)$ and $\phi(P_3)$) or decrease monotonically (see $\phi(P_1)$ and $\phi(P_2)$) when more items are appended to a given pattern. In our preliminary work [Tang et al. 2012], we focused on harmonic occupancy. In case that in some applications the arithmetic average is required, we focus on arithmetic occupancy in this study for technical complement to our previous work [Tang et al. 2012]. Without loss of generality, in the following we use $\phi(P)$ to represent $\phi_A(P)$.

Note that the theoretically comparison of arithmetic occupancy and harmonic occupancy is actually equal to compare the two average functions (i.e., arithmetic average and harmonic average) for different applications. In fact, the arithmetic average is best used in situations in which the data are not skewed (no extreme outliers). Accordingly, the harmonic average is best to use when there is: a large population in which the majority of the values are distributed uniformly but where there are a few outliers with significantly higher values [Haff 1979].

Definition 2.4 (*Dominant Pattern*). For a given minimum occupancy threshold β $(0 < \beta \leq 1)$, the pattern P is said to be *dominant* if $\phi(P) \geq \beta$.

With the definition of support and occupancy, we can measure the *quality* of a pattern by combining these two factors.

Definition 2.5 (*Quality*). The *quality* (value) of a pattern P is defined as $q(P) = \mathcal{Q}(\sigma(P), \phi(P))$, where $\mathcal{Q}(\sigma(P), \phi(P))$ is any function, which maps the support and occupancy to a real value.

In this article, we use the weighted sum function, that is, $q(P) = \sigma(P) + \lambda\phi(P)$, where the weight λ $(0 \leq \lambda < +\infty)$ is a user defined parameter to capture the relative importance of support and occupancy. It is worth mentioning that any other functions, such as the harmonic average (similar to the F1 score) and the sum of logarithms (similar to block size proposed in Gade et al. [2004]) can be used to combine the two values of support and occupancy. Later, we will see that the proposed techniques can be applied to any function $\mathcal{Q}(\sigma(P), \phi(P))$, which is monotonically increasing with respect to $\phi(P)$.

2.2. Problem Formulation

Mining Qualified Patterns. Given the minimal support threshold α and the minimal occupancy threshold β, the task is to find all the qualified patterns (which are both frequent and dominant) in a database $\mathcal{D}$.

In addition, among all the qualified patterns we also aim to find the pattern with the maximal quality value for recommendation. Formally, it can be formulated as follows.

Mining Top Qualified Pattern. The top qualified pattern P is defined as the qualified pattern with the maximal quality value:

$$\mathop{\arg\max}_{P:\sigma(P)\geq\alpha,\phi(P)\geq\beta} \{\sigma(P) + \lambda\phi(P)\}. \tag{1}$$

Later, in Section 3 we will show that the top qualified pattern must be closed one, thus the proposed algorithm DOFRA for top qualified pattern mining can be enhanced by using the pruning methods for closed patterns for efficient mining.

There are three parameters in the formulation of mining top qualified pattern, namely α, β, λ. According to α, β, if there is no pattern that is both frequent and dominant, the top qualified pattern does not exist and a valid algorithm will not output any result since no non-empty pattern exists that meets the quality requirements. The parameter λ is a user defined parameter to capture the relative importance of support and occupancy.

In the remaining of the article, we will primarily focus on mining top (i.e., $k = 1$) qualified pattern. We then show that the solution to top qualified pattern mining can be easily extended to solve the problem of identifying top-k qualified patterns for $k > 1$.

2.3. Discussion

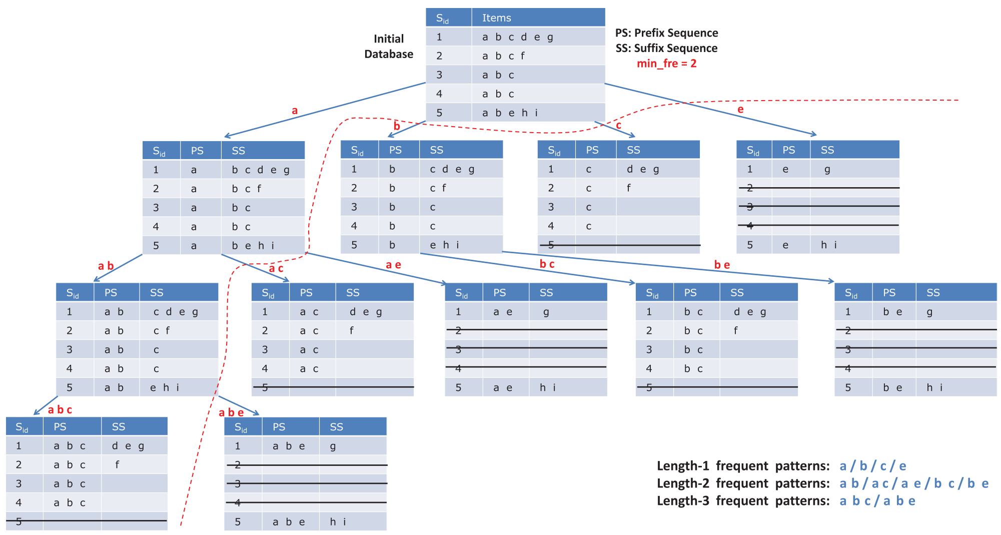
One may think that dominant patterns with high occupancy usually contain a large number of items and thus methods based on *Maximal Frequent Pattern* mining may be adopted for identifying top-k qualified patterns. (A pattern P is a maximal frequent pattern if P is frequent and no super-pattern of P is frequent [Burdick et al. 2001]). Given a minimal support threshold we can get multiple maximal frequent patterns, among them we can select the one with the largest number of items as the top qualified pattern. Is this a valid algorithm for mining top qualified pattern? The answer is “no” for the following reasons.

First, in methods based on mining maximal frequent patterns, the number of items in a pattern is used as a measure for pattern selection. Compared with the definition of occupancy, this is actually the *absolute* size of a pattern while occupancy is the *relative* size of a pattern which is the average ratio of the size of the pattern to the number of items in its supporting sequence. Among all the frequent patterns, the support of maximal frequent pattern selected is usually lower than that of our proposed method, thus its precision is lower than the proposed method. Secondly, note that there may exist many different maximal patterns with the same largest length, in which it is difficult to choose one for recommendation by only considering the absolute size of a pattern. In other words, choosing different patterns randomly may occur very different final performance. Lastly, in mining top qualified pattern a weighted sum of both support and occupancy is used as the interestingness measure, which may lead to better recommendation performance compared to the patterns selected by methods based on maximal frequent patterns. The experimental results in Section 5 further validate our analysis here.

In the next section we will present our general algorithm, named DOFRA, for the top qualified pattern mining problem.

3. OVERVIEW OF DOFRA

The straightforward solution to this pattern mining problem is to first generate all the frequent patterns, calculate the occupancy and quality value for each pattern, and then select the qualified pattern with maximum quality value. However, it is time-consuming to calculate the occupancy value for each frequent pattern. In this section, we will show how the properties on occupancy and quality measures can be injected deeply into the search process and greatly prune the search space. Before going further,


Initial Database
PS: Prefix Sequence
SS: Suffix Sequence
min_fre = 2
a
b
c
e
a b
a c
a e
b c
b e
a b c
a b e
Length-1 frequent patterns: a / b / c / e
Length-2 frequent patterns: a b / a c / a e / b c / b e
Length-3 frequent patterns: a b c / a b e

S_{id}	Items
1	a b c d e g
2	a b c f
3	a b c
4	a b c
5	a b e h i

S_{id}	PS	SS
1	a	b c d e g
2	a	b c f
3	a	b c
4	a	b c
5	a	b e h i

S_{id}	PS	SS
1	b	c d e g
2	b	c f
3	b	c
4	b	c
5	b	e h i

S_{id}	PS	SS
1	c	d e g
2	c	f
3	c	
4	c	
~~5~~		

S_{id}	PS	SS
1	e	g
~~2~~		
~~3~~		
~~4~~		
5	e	h i

S_{id}	PS	SS
1	a b	c d e g
2	a b	c f
3	a b	c
4	a b	c
5	a b	e h i

S_{id}	PS	SS
1	a c	d e g
2	a c	f
3	a c	
4	a c	
~~5~~		

S_{id}	PS	SS
1	a e	g
~~2~~		
~~3~~		
~~4~~		
5	a e	h i

S_{id}	PS	SS
1	b c	d e g
2	b c	f
3	b c	
4	b c	
~~5~~		

S_{id}	PS	SS
1	b e	g
~~2~~		
~~3~~		
~~4~~		
5	b e	h i

S_{id}	PS	SS
1	a b c	d e g
2	a b c	f
3	a b c	
4	a b c	
~~5~~		

S_{id}	PS	SS
1	a b e	g
~~2~~		
~~3~~		
~~4~~		
5	a b e	h i

Fig. 3. Construction of prefix-projected database tree for the database shown in Table I.

we first introduce a typical sequential pattern mining algorithm which is related to our work.

3.1. Basic Idea

Pattern mining algorithms on a sequence database usually adopt PrefixSpan [Pei et al. 2001] to find the frequent patterns. PrefixSpan works in a divide-and-conquer way. After the first scan of the database, the set of length-1 frequent patterns is derived. Each pattern is treated as a prefix and the complete set of frequent patterns can be partitioned into different subsets according to different prefixes. To mine the subsets of frequent patterns, corresponding projected databases are constructed and mined recursively.

Figure 3 shows a prefix-projected database tree constructed by a revised PrefixSpan[3] on the database shown in Table I. Each node in this tree is a prefix (pattern) projected database which contains three parts: the supporting sequence S_{id}, the appearance PS (i.e., prefix sequence) of this pattern and the remaining sequence SS (i.e., suffix sequence) behind this pattern in the supporting sequence. For *bc*-projected database in Figure 3 as an example, the prefix sequence and suffix sequence of *bc* in S_1 is *bc* and *deg*, respectively. After scanning the initial database, we get length-1 frequent patterns *a*/*b*/*c*/*e* and each pattern can be treated as a prefix and its corresponding projected database is then constructed. Next, we recursively consider these projected databases until no frequent items exist in the suffix sequences. For example, in the b-projected database in Figure 3, the items c and e are frequent in the suffix sequences, thus b-projected database can be further expanded to bc- and be-projected databases. Because there are no frequent items in the bc's supporting suffix sequences, thus this projected database ceased to expand. The traversal in this database tree is to find all frequent patterns.

Next we will show our solution exploring the properties on occupancy and quality to further prune the search space in this tree. Specifically, given a pattern P we can

[3]Note that the step I-extension [Ayres et al. 2002] is removed in PrefixSpan for frequent *single-itemset* sequences enumeration.

estimate the upper bounds of occupancy and quality for all frequent patterns in the subtree rooted at P. In other words, the occupancy and quality of any pattern in the given subtree will be no bigger than its upper bounds, respectively. If the upper bound on occupancy is smaller than the minimal occupancy threshold β, this subtree should be pruned. Also, we can maintain the current biggest quality value in the search process so far, denoted by q^*, and all subtrees with the quality upper bounds less than q^* should be pruned.

Take the subtree rooted at pattern b in Figure 3 as an example. When the node b is searched, we can give an upper bound of the quality for all frequent patterns in this subtree (including b, bc and be) and this upper bound is 2.05 (the fast computation for this quality upper bound is shown in Section 4). Assume that we have already found abc is the node with the highest quality 2.42 so far, which is bigger than the upper bound for the subtree rooted at b. Thus, the subtree rooted at b should be pruned. The dashed line in Figure 3 is the cut line for pruning the search space for mining top qualified pattern ($\alpha = 0.4$, $\beta = 0.5$, and $\lambda = 2$.). In this example, the search space when using only the frequency constraint for pruning has 12 projected databases while the search space of DOFRA has only 4 projected databases. The search space is greatly reduced by pruning using the upper bounds of occupancy and quality. To further reduce the search space, we have the following property.

PROPERTY 1 (CLOSED PATTERN MAXIMIZES OCCUPANCY). *A pattern P is* closed *if P is frequent and there exists no proper super-pattern of P with the same support. All sub-patterns of the closed pattern P with the same support form an equivalence class of P. Among this equivalence class, the closed pattern P must be the one with maximal occupancy value.*

Given the toy example in Table I, pattern abc is a closed pattern with support 0.8 (suppose $\alpha = 0.5$). The equivalence class of abc is $\{c, bc, abc\}$. Their occupancy values are 0.27, 0.54, and 0.81, respectively. Among this equivalence class, abc has the biggest occupancy value. According to the Property 1, the top qualified pattern must be closed. In other words, any nonclosed pattern must not be the top qualified one. In other words, if there are no closed patterns in the subtree rooted at P, this subtree should be pruned. Thus, we adopt the effective technique of *BackScan search space pruning* proposed in BIDE [Wang and Han 2004] for the subtree closeness checking (readers can refer [Wang and Han 2004] for more details).

3.2. The Algorithm

Algorithm 1 shows the pattern mining process of a Depth-First-Search (DFS) with the pruning techniques based on the framework shown in Figure 3. Note that since each node in the search tree uniquely corresponds to a pattern, we use the terms "node" and "pattern" interchangeably here. At current node *currNode*, we first compare its quality value with the top qualified node searched so far. If the quality value of *currNode* is larger than that of top qualified node (line 1), we use *currNode* to replace the top qualified node (line 2). Then, we check any child, denoted by *node*, of *currNode*. If the support of *node* is smaller than minimal support threshold α (line 5), or its occupancy upper bound is smaller than minimal occupancy threshold β (line 7), or its quality upper bound is smaller than the current maximal quality value (line 9) or there are no closed patterns in the subtree rooted at *node* (line 11), then *node* should be pruned. Otherwise, we recursively check *node* (line 12).

Based on the previous algorithm, we can easily extend it to solve the problem of identifying top-k qualified (closed) patterns. Specifically, by replacing *bestNode* in the input, a usual way is to use heap H with size k to maintain the top-k qualified nodes searched so far [Han et al. 2002; Wu et al. 2012]. In line 1, the *bestNode.quality* should

Table II. The Supporting Sequences of *b* in Table I

S_{id}	S	PS	SS	|S|	|PS|	|SS|
S_1	a b c d e g	b	c d e g	6	1	4
S_2	a b c f	b	c f	4	1	2
S_3	a b c	b	c	3	1	1
S_4	a b c	b	c	3	1	1
S_5	a b e f i	b	e f i	5	1	3
				SL	PSL	SSL

ALGORITHM 1: DOFRA

```
Input: the current Node currNode obtained from the prefix-projected database tree like
       Figure 3, the top qualified pattern searched so far: bestNode.
1  if $currNode.quality \geq bestNode.quality$ then
2      $bestNode \leftarrow currNode$;
3  for $node \in currNode.children$ do
4      $supp \leftarrow node.support$;
5      if $(supp \geq \alpha)$ then
6          $occu \leftarrow$ the occupancy bound on the subtree rooted at node;
7          if $(occu \geq \beta)$ then
8              $qual \leftarrow$ the quality bound on the subtree rooted at node;
9              if $(qual \geq bestNode.quality)$ then
10                 $clos \leftarrow$ true if there exists closed patterns in the subtree rooted at node and
                   vice verse;
11                 if $(clos == true)$ then
12                     DOFRA (node, bestNode);
```

be replaced by $H_k.quality$ (H_k is the k-th node with minimal quality value in H), then *bestNode* in line 2 should be replaced by H_k. At last, we use $H_k.quality$ to replace *bestNode.quality* in line 9.

So far we have omitted the most challenging part in the algorithm: how to compute the upper bounds on occupancy and quality (line 6 and line 8). Since the estimation of these upper bounds is at the cost of extra computing, they should be computed very efficiently. Meanwhile, to prune the search space as much as possible, it is required that these upper bounds should be as tight as possible. In the next section, we will describe its details.

4. THE UPPER BOUNDS OF OCCUPANCY AND QUALITY

In this section, we first give an overview of the upper bounds estimation. Then we show how to efficiently compute the upper bounds of the arithmetic occupancy and its quality. For more details about harmonic occupancy and its upper bounds, readers can refer to our preliminary work [Tang et al. 2012].

4.1. The Overview of the Upper Bound

For any subtree, let P be the root pattern (node) and $\mathcal{D}_P$ be the supporting sequences of P. For any sequence S in $\mathcal{D}_P$, the *Prefix Sequence* (*PS* for short) of P in S is the appearance of P in S while its *Suffix Sequence* (*SS* for short) is the remaining sequence following P in S. For the supporting sequences of pattern b shown in Table II, the prefix sequence and suffix sequence of b in S_1 is b and $cdeg$, respectively. Table II shows all prefix sequences and suffix sequences of b in its supporting sequences. To compute the upper bounds, we need the following notions with respect to P.

Table III. Notations used in Section 4

P	The root pattern for a subtree
P'	Any pattern in the subtree of P
$\mathcal{D}_P$	The supporting sequences of P in $\mathcal{D}$
$freq_{min}$	The minimal frequency threshold ($\lceil \alpha \cdot \lvert\mathcal{D}\rvert \rceil$)
u	u often denotes $\lvert\mathcal{D}_{P'}\rvert$
v	v often denotes $\lvert P'\rvert - \lvert P\rvert$
PS	The prefix sequence of P in S
SS	The suffix sequence of P in S
PSL	Prefix sequence length vector, $PSL[i] = \lvert PS_i\rvert$
SSL	Suffix sequence length vector, $SSL[i] = \lvert SS_i\rvert$
SL	Sequence length vector, $SL[i] = \lvert S_i\rvert$
$\mathsf{V}^{\uparrow/\downarrow}$	vector V sorted by ascending/descending order

Definition 4.1 (*SL & PSL & SSL*). The Sequence Length vector of P, denoted as SL, is used to record the number of items in its supporting sequence S ($S \in \mathcal{D}_P$). The Prefix Sequence Length vector of P, denoted as PSL, is used to record the number of items in each prefix sequence in S ($S \in \mathcal{D}_P$). The Suffix Sequence Length vector of P, denoted as SSL, is used to record the number of items in each suffix sequence in S ($S \in \mathcal{D}_P$).

SL, PSL, and SSL are three vectors. Their index refers to the sequence number. As shown in Table II, $\mathsf{PSL}[5] = 1$, $\mathsf{SSL}[5] = 3$, and $\mathsf{SL}[5] = 5$. Note that $\mathsf{PSL}[i] + \mathsf{SSL}[i] \leq \mathsf{SL}[i]$.

In the following, we will use u to denote the frequency of a pattern $P'(\lvert\mathcal{D}_{P'}\rvert = u)$ in the subtree rooted at P, and v to denote extension length of P' with respect to P ($\lvert P'\rvert - \lvert P\rvert = v$). For example, suppose $P = b$ and $P' = bcdef$, the extension length of P' with respect to P is 4. In addition, we will need to frequently sort the values in a vector. For a vector V, $\mathsf{V}^{\downarrow}(\mathsf{V}^{\uparrow})$ is the vector obtained by sorting V in the descending (ascending) order of the values. Thus, $\mathsf{V}^{\downarrow}(i)$ is the ith largest value in V. For the SSL in Table II as an example, $\mathsf{SSL}^{\downarrow} = \langle 4, 3, 2, 1, 1\rangle$ and $\mathsf{SSL}^{\downarrow}[2] = 3$. The notations used in this section are summarized in Table III.

We aim to estimate the occupancy and quality upper bounds of all nodes in the subtree of P. The basic idea is briefly described as follows.

First, assume that we have already known the frequency u of any pattern P' in the subtree of P. Then, we will propose a function $\mathcal{F}(u, \mathsf{PSL}, \mathsf{SSL}, \mathsf{SL})$ satisfying the following conditions:

$$\phi(P') \leq \mathcal{F}(u, \mathsf{PSL}, \mathsf{SSL}, \mathsf{SL}). \tag{2}$$

It is worth mentioning that the computing of $\mathcal{F}$ only involves u, PSL, SSL, SL.

Then, we will show $\mathcal{F}(u, \mathsf{PSL}, \mathsf{SSL}, \mathsf{SL})$ is not increasing with the increase of u, namely

$$\mathcal{F}(u+1, \mathsf{PSL}, \mathsf{SSL}, \mathsf{SL}) \leq \mathcal{F}(u, \mathsf{PSL}, \mathsf{SSL}, \mathsf{SL}). \tag{3}$$

We call in Equation (3) the *anti-monotonicity* property of occupancy upper bound w.r.t. u.

Since P' is a frequent super-pattern of P, the range of u is $[freq_{min}, \lvert\mathcal{D}_P\rvert]$, where $freq_{min}$ is the minimal frequency threshold ($freq_{min} = \lceil \alpha \cdot \lvert\mathcal{D}\rvert \rceil$). Thus, we have the following theorems.

THEOREM 4.2 (OCCUPANCY UPPER BOUND). *For any pattern P' in the subtree of P*

$$\phi(P') \leq \mathcal{F}(freq_{min}, \mathit{PSL}, \mathit{SSL}, \mathit{SL}). \tag{4}$$

PROOF. The conclusion holds if $\mathcal{F}(u, \mathsf{PSL}, \mathsf{SSL}, \mathsf{SL})$ satisfies Inequalities (2) and (3). □

Table IV. The Procedure for Computing F(3, PSL, SSL, SL)

S_{id}	SL	PSL	SSL	Step (1)	Step (2)
S_1	6	1	4	(1+4)/6	✓
S_2	4	1	2	(1+2)/4	✓
S_3	3	1	1	(1+1)/3	
S_4	3	1	1	(1+1)/3	
S_5	5	1	3	(1+3)/5	✓

Note that in Theorem 4.2 it is not required to know the frequency of P'. Thus, this is the occupancy upper bound of any node in the subtree of P. Next, for the upper bound of quality we also have

Theorem 4.3 (Quality Upper Bound). *For any pattern P' in the subtree of P*

$$q(P') \leq \max_{freq_{min} \leq u' \leq |\mathcal{D}_P|} \mathcal{Q}\left(\frac{u'}{|\mathcal{D}|}, \mathcal{F}(u', PSL, SSL, SL)\right), \tag{5}$$

when $\mathcal{Q}(A, B)$ is monotonically increasing w.r.t B.

Proof. Since $\mathcal{Q}(A, B)$ is monotonically increasing w.r.t B, thus for any P' in the subtree of P with the frequency u ($|\mathcal{D}_{P'}| = u$),

$$q(P') = \mathcal{Q}\left(\frac{u}{|\mathcal{D}|}, \phi(P')\right) \leq \mathcal{Q}\left(\frac{u}{|\mathcal{D}|}, \mathcal{F}(u, \text{PSL}, \text{SSL}, \text{SL})\right). \tag{6}$$

Then, the conclusion holds if $\mathcal{F}(u, \text{PSL}, \text{SSL}, \text{SL})$ satisfies Inequality (2). □

Theorem 4.3 gives the quality upper bound of any node in the subtree of P. In this article, we use the weighted sum of support and occupancy, thus the quality upper bound is

$$\max_{freq_{min} \leq u' \leq |\mathcal{D}_P|} \frac{u'}{|\mathcal{D}|} + \lambda \cdot \mathcal{F}(u', \text{PSL}, \text{SSL}, \text{SL}). \tag{7}$$

In the following, we will propose the $\mathcal{F}$ functions which satisfy Inequalities (2) and (3).

4.2. The Upper Bounds of Occupancy and Quality

In this subsection, we will propose two instances of F which satisfy Inequalities (2) and (3). We will show that the first F is more efficient, however, less tight than the second one. We also theoretically prove that the second F gives the tightest upper bound if only the values of PSL, SSL, SL are used for the computation. Finally, we show how to achieve the tradeoff between the bound tightness and computational efficiency.

4.2.1. The Efficient Upper Bound $F(u, PSL, SSL, SL)$. We give the following example to show how the F(u, PSL, SSL, SL) is developed. For the data of PSL, SSL, SL in Table II, we aim to find the occupancy upper bound of any frequent pattern P' (suppose its frequency $u = 3$) in the subtree rooted at b. The procedure is as follows.

(1) In order to get the maximal occupancy value of P' in each sequence, we can use the length of the suffix sequence as the extension length of P'. For example, for the sequence S_1, the maximal occupancy value of P' in this sequence is $(1 + 4)/6$. We conduct this process for each sequence and get its possibly maximal occupancy value as shown in the fifth column of Table IV.

(2) Since the frequency of P' is 3, from the 5 sequences we need to identify 3 of them with the top-3 biggest occupancy values. Thus the sequences of S_1, S_2, S_5 are selected. By averaging these three values we get this upper bound, that is, $F(3, \text{PSL}, \text{SSL}, \text{SL}) = 1/3 \times (4/5 + 3/4 + 5/6) \approx 0.79$.

By translating the previous procedure into math presentation, we propose $F(u, \mathsf{PSL}, \mathsf{SSL}, \mathsf{SL})$ function as follows.

$$F(u, \mathsf{PSL}, \mathsf{SSL}, \mathsf{SL}) = \frac{1}{u} \cdot \max_{l_1,\ldots,l_u} \sum_{i=1}^{u} \frac{\mathsf{PSL}[l_i] + \mathsf{SSL}[l_i]}{\mathsf{SL}[l_i]}, \tag{8}$$

where l_i is the index of any supporting sequence of P in SL/PSL/SSL and $\max_{l_1,\ldots,l_u} M$ means that u sequences are selected from the supporting sequences of P to maximize any function M.

Next, we propose Properties 2 and 3 to show that the F function in Equation (8) satisfies Inequalities (2) and (3).

PROPERTY 2. *For any P' in the subtree of P, let u be the frequency of P'. Then, the occupancy of P' satisfies that*

$$\phi(P') \leq F(u, \mathit{PSL}, \mathit{SSL}, \mathit{SL}). \tag{9}$$

PROOF. Let's consider the occupancy of P'.

$$\phi(P') = \frac{1}{u} \sum_{S \in \mathcal{D}_{P'}} \frac{|P'|}{|S|} \tag{10}$$

$$= \frac{1}{u} \sum_{S \in \mathcal{D}_{P'}} \frac{|P| + |P'| - |P|}{|S|} \tag{11}$$

$$\leq \frac{1}{u} \sum_{S \in \mathcal{D}_{P'}} \frac{\mathsf{PSL}[l_S] + \mathsf{SSL}[l_S]}{\mathsf{SL}[l_S]} \tag{12}$$

$$\leq \frac{1}{u} \max_{l_1,\ldots,l_u} \sum_{i=1}^{u} \frac{\mathsf{PSL}[l_i] + \mathsf{SSL}[l_i]}{\mathsf{SL}[l_i]} \tag{13}$$

Note that l_S is the corresponding index of S in SL/PSL/SSL and Inequality (12) is due to $(|P'| - |P|) \leq \mathsf{SL}[l_S]$. Since $P \sqsubseteq P'$, by the anti-monotonicity of frequent pattern, $\mathcal{D}_{P'} \subseteq \mathcal{D}_P$, Inequality (13) holds. □

PROPERTY 3. $F(u+1, \mathit{PSL}, \mathit{SSL}, \mathit{SL}) \leq F(u, \mathit{PSL}, \mathit{SSL}, \mathit{SL})$.

PROOF. Set $A_i = \max_{l_i} \frac{\mathsf{PSL}[l_i]+\mathsf{SSL}[l_i]}{\mathsf{SL}[l_i]}$ (A_i is sorted by descending order)

$$F(u+1, \mathsf{PSL}, \mathsf{SSL}, \mathsf{SL}) - F(u, \mathsf{PSL}, \mathsf{SSL}, \mathsf{SL}) \tag{14}$$

$$= \frac{1}{u+1} \sum_{i=1}^{u+1} A_i - \frac{1}{u} \sum_{i=1}^{u} A_i \tag{15}$$

$$= \frac{1}{u+1} \left(\sum_{i=1}^{u} A_i + A_{u+1} \right) - \frac{1}{u} \sum_{i=1}^{u} A_i \tag{16}$$

$$= \frac{1}{u+1} \left(A_{u+1} - \frac{1}{u} \sum_{i=1}^{u} A_i \right) \leq 0 \tag{17}$$

It is easy to check that $A_{u+1} - \frac{1}{u} \sum_{i=1}^{u} A_i \leq 0$ since $A_u \geq A_{u+1} \geq 0$, Inequality (17) holds. □

With Properties 2 and 3 we can also easily prove that the results in Theorems 4.2 and 4.3 hold for $F(u, \mathsf{PSL}, \mathsf{SSL}, \mathsf{SL})$.

The following example shows how the occupancy upper bound is computed for the data in Table II (Suppose $freq_{min} = 2$). According to the function F proposed in Equation (8) and Theorem 4.2, the occupancy upper bound for the subtree rooted at b, denoted as $\widehat{\phi}(b)$, is

$$\widehat{\phi}(b) = F(2, \mathsf{PSL}, \mathsf{SSL}, \mathsf{SL}) = 1/2 \times (4/5 + 5/6) \approx 0.82.$$

It means that the occupancy of all frequent patterns in this subtree, including pattern b, bc and be, are less than 0.82.

ALGORITHM 2: The quality upper bound based on $F(u, \mathsf{PSL}, \mathsf{SSL}, \mathsf{SL})$

```
  input: the root pattern $P$, the corresponding three vectors PSL, SSL, SL and $n = |\mathcal{D}_P|$.
  output: $qual$, the quality upper bound of any frequent pattern in the subtree rooted at $P$.
1 for $i \leftarrow 1$ to $n$ do
2     $A(i) \leftarrow (\mathsf{PSL}[i] + \mathsf{SSL}[i])/\mathsf{SL}[i]$;
3 sort $A$ by descending order;
4 $qual \leftarrow 0, occu \leftarrow 0, sum \leftarrow 0$;
5 for $u \leftarrow 1$ to $n$ do
6     $sum \leftarrow sum + A^{\downarrow}(u)$;
7     if $u \geq freq_{min}$ then
8         $occu = sum/u$;
9         $qual = \max(qual, u/|\mathcal{D}| + \lambda \cdot occu)$ ;
```

Algorithm 2 gives the pseudocode for computing the quality upper bound in Theorem 4.3 with $F(u, \mathsf{PSL}, \mathsf{SSL}, \mathsf{SL})$. The complexity of Algorithm 2 is $O(n \cdot log(n))$. We first use the $O(n)$ time for computing A for the loop in line 1 and then the $O(n \cdot log(n))$ time in line 3 for sorting A with *quick sort algorithm*, finally $O(n)$ time for the loop in line 5.

4.2.2. The "Tightest" Upper Bound $F'(u, \mathsf{PSL}, \mathsf{SSL}, \mathsf{SL})$. As mentioned before, $F(u, \mathsf{PSL}, \mathsf{SSL}, \mathsf{SL})$ has the parameter u, namely the frequency for any pattern P' in the subtree of P. Using another new parameter $v(|P'| - |P|)$, namely the extension length of P' with respect to P, we may obtain a stronger upper bound.

The basic idea to this stronger bound is as follows: (1) Let P' be any frequent pattern in the subtree rooted at P with the frequency u ($|\mathcal{D}_{P'}|$) and the extension length v ($|P'| - |P|$), then we can propose a function $F'(u, v, \mathsf{PSL}, \mathsf{SSL}, \mathsf{SL})$ s.t. $\phi(P') \leq F'(u, v, \mathsf{PSL}, \mathsf{SSL}, \mathsf{SL})$. (2) We enumerate all possible v to get the final maximal value (i.e., upper bound) for P', namely $F'(u, \mathsf{PSL}, \mathsf{SSL}, \mathsf{SL})$.

The key procedure to this basic idea is how to propose $F'(u, v, \mathsf{PSL}, \mathsf{SSL}, \mathsf{SL})$. Next, we will give the following example to show how the $F'(u, v, \mathsf{PSL}, \mathsf{SSL}, \mathsf{SL})$ is developed. For the data of PSL, SSL, SL in Table II, we aim to find the occupancy upper bound of any frequent pattern P' in the subtree rooted at b with the frequency $u = 2$ and the extension length $v = 2$. The procedure is as follows.

(1) Identify the sequences whose suffix sequences contains at least two items. Otherwise, the sequences must not contain P'. Thus, three sequences of S_1, S_2, S_5 are selected while sequences S_3, S_4 are filtered out.

(2) In order to get the maximal occupancy value of P' in each sequence, we can directly use the extension length v. For example, for the sequence S_1, the maximal occupancy value P' in this sequence is $(1 + 2)/6$. We conduct this process for each selected sequence from Step (1).

(3) Lastly, from the three sequences we need to identify two of them with the top-2 biggest occupancy values. This time the sequences of S_2, S_5 are selected. By averaging

these two values we get this upper bound, that is, $F(2, 2, \mathsf{PSL}, \mathsf{SSL}, \mathsf{SL}) = 1/2 \times (3/4 + 3/5) \approx 0.68$.

By translating the earlier procedure into math presentation, we first propose $F'(u, v, \mathsf{PSL}, \mathsf{SSL}, \mathsf{SL})$ and then give $F'(u, \mathsf{PSL}, \mathsf{SSL}, \mathsf{SL})$ with the following properties.

Property 4. *Let* $F'(u, v, \mathit{PSL}, \mathit{SSL}, \mathit{SL})$

$$= \frac{1}{u} \cdot \max_{\substack{l_1,\ldots,l_u,\\ \mathit{SSL}[l_i] \geq v}} \sum_{i=1}^{u} \frac{\mathit{PSL}[l_i] + v}{\mathit{SL}[l_i]}. \tag{18}$$

Then for any P' in the subtree of P with the frequency u and the extension length v, we have $\phi(P') \leq F'(u, v, \mathit{PSL}, \mathit{SSL}, \mathit{SL})$. Furthermore, for any pattern P' in the subtree of P with the frequency u (and no constraint on v), we have

$$\phi(P') \leq \max_{0 \leq v \leq \mathit{SSL}^{\downarrow}[u]} F'(u, v, \mathit{PSL}, \mathit{SSL}, \mathit{SL}) \tag{19}$$

$$\triangleq F'(u, \mathit{PSL}, \mathit{SSL}, \mathit{SL}) \tag{20}$$

Proof. For any pattern P' in the subtree of P, we have

$$\phi(P') = \frac{1}{u} \sum_{S \in \mathcal{D}_{P'}} \frac{|P'|}{|S|} \tag{21}$$

$$= \frac{1}{u} \sum_{S \in \mathcal{D}_{P'}} \frac{|P| + |P'| - |P|}{|S|} \tag{22}$$

$$= \frac{1}{u} \sum_{S \in \mathcal{D}_{P'}} \frac{\mathsf{SSL}[l_S] + v}{\mathsf{SL}[l_S]} \tag{23}$$

$$\leq \frac{1}{u} \cdot \max_{\substack{l_1,\ldots,l_u,\\ \mathsf{SSL}[l_i] \geq v}} \sum_{i=1}^{u} \frac{\mathsf{PSL}[l_i]) + v}{\mathsf{SL}[l_i]}. \tag{24}$$

Since $P \sqsubseteq P'$, by the anti-monotonicity of frequent pattern, $\mathcal{D}_{P'} \subseteq \mathcal{D}_P$. Then we can assign any v item labels appended to P and select u sequences from $\mathcal{D}_P$ to get a maximal occupancy value for P'. Note that the length of the suffix sequence of P in any of the selected u sequences should be no less than v (i.e., $\mathsf{SSL}[l_i] \geq v$), otherwise, the sequence must not contain P'. Thus, in Equation (24) holds.

Then we enumerate all possible v to get the final maximal value for P', namely $F'(u, \mathsf{PSL}, \mathsf{SSL}, \mathsf{SL})$. Note that the largest valid extension length of P' with respect to P should be no larger than $\mathsf{SSL}^{\downarrow}[u]$, otherwise, P' must not be frequent. We unify the bounds F' over v and then get

$$\phi(P') \leq \max_{0 \leq v \leq \mathsf{SSL}^{\downarrow}[u]} F'(u, v, \mathsf{PSL}, \mathsf{SSL}, \mathsf{SL}). \quad \square$$

Property 5. $F'(u+1, \mathit{PSL}, \mathit{SSL}, \mathit{SL}) \leq F'(u, \mathit{PSL}, \mathit{SSL}, \mathit{SL})$. *The definition of $F'(u, \mathit{PSL}, \mathit{SSL}, \mathit{SL})$ is given in Equation (20).*

Proof. By the definition of F', similar to the proof of Property 3, we have $F'(u+1, v, \mathsf{PSL}, \mathsf{SSL}, \mathsf{SL}) \leq F'(u, v, \mathsf{PSL}, \mathsf{SSL}, \mathsf{SL})$ for any u, v. Since $\mathsf{SSL}^{\downarrow}[u+1] \leq \mathsf{SSL}^{\downarrow}[u]$, we have

$$F'(u+1, \mathsf{PSL}, \mathsf{SSL}, \mathsf{SL}) = \max_{0 \leq v \leq \mathsf{SSL}^{\downarrow}[u+1]} F'(u+1, v, \mathsf{PSL}, \mathsf{SSL}, \mathsf{SL}) \tag{25}$$

$$\leq \max_{0 \leq v \leq \mathsf{SSL}^{\downarrow}[u+1]} F'(u, v, \mathsf{PSL}, \mathsf{SSL}, \mathsf{SL}) \tag{26}$$

$$\leq \max_{0 \leq v \leq \mathsf{SSL}^{\downarrow}[u]} F'(u, v, \mathsf{PSL}, \mathsf{SSL}, \mathsf{SL}) \tag{27}$$

$$= F'(u, \mathsf{PSL}, \mathsf{SSL}, \mathsf{SL}). \quad \square \tag{28}$$

With Properties 4 and 5 we can also easily prove that the results in Theorems 4.2 and 4.3 hold for $F'(u, \mathsf{PSL}, \mathsf{SSL}, \mathsf{SL})$.

The following example shows how the occupancy upper bound is computed for the data in Table II. We can sort SSL by descending order, that is, $\mathsf{SSL}^{\downarrow} = \langle 4, 3, 2, 1, 1 \rangle$. Suppose $freq_{min} = 2$, then $\mathsf{SSL}^{\downarrow}[freq_{min}] = 3$. According to the function F' proposed in Equation (20) and Theorem 4.2, the occupancy upper bound for the subtree rooted at b is

$$\widehat{\phi}(b) = \max_{0 \leq v \leq 3} F'(2, v, \mathsf{PSL}, \mathsf{SSL}, \mathsf{SL}).$$

According to Equation (18), we can get

$$\begin{aligned} F'(2, 0, \mathsf{PSL}, \mathsf{SSL}, \mathsf{SL}) &= 1/2 \times (1/3 + 1/3) \approx 0.33 \\ F'(2, 1, \mathsf{PSL}, \mathsf{SSL}, \mathsf{SL}) &= 1/2 \times (2/3 + 2/3) \approx 0.67 \\ F'(2, 2, \mathsf{PSL}, \mathsf{SSL}, \mathsf{SL}) &= 1/2 \times (3/4 + 3/5) \approx 0.68 \\ F'(2, 3, \mathsf{PSL}, \mathsf{SSL}, \mathsf{SL}) &= 1/2 \times (4/6 + 4/5) \approx 0.73. \end{aligned}$$

Thus $\widehat{\phi}(b) = max\{0.33, 0.67, 0.68, 0.73\} = 0.73$, which is smaller than 0.82 (the upper bound based on function F in Equation (8)). In other words, F' is tighter than F.

Algorithm 3 gives the pseudocode for computing the quality upper bound in Theorem 4.3 with $F'(u, v, \mathsf{PSL}, \mathsf{SSL}, \mathsf{SL})$. The complexity of Algorithm 3 is $O(n \cdot log(n) \cdot \mathsf{SSL}^{\downarrow}[freq_{min}])$. We first use $O(n \cdot log(n))$ time for sorting SSL with *quick sort algorithm* in line 1, then use $O(n \cdot log(n) \cdot \mathsf{SSL}^{\downarrow}[freq_{min}])$ time for the double loop from line 3 to line 8.

ALGORITHM 3: The quality upper bound based on $F'(u, v, \mathsf{PSL}, \mathsf{SSL}, \mathsf{SL})$

```
   input: the root pattern $P$ and its corresponding three vectors PSL, SSL, SL and $n = |\mathcal{D}_P|$.
   output: $qual$, the quality upper bound of any pattern in the subtree rooted at $P$.
1  sort SSL by descending order;
2  $qual \leftarrow 0, occu \leftarrow 0, sum \leftarrow 0$;
3  for $v \leftarrow 0$ to $SSL^{\downarrow}[freq_{min}]$ do
4      for $i \leftarrow 1$ to $n$ do
5          if $SSL[i] \geq v$ then
6              $A(i) \leftarrow (\mathsf{PSL}[i] + v)/\mathsf{SL}[i]$;
7      sort $A$ by descending order;
8      for $u \leftarrow 1$ to $A.size()$ do
9          $sum \leftarrow sum + A^{\downarrow}(u)$;
10         if $u \geq freq_{min}$ then
11             $occu = sum/u$;
12             $qual = \max(qual, u/|\mathcal{D}| + \lambda \cdot occu)$;
```

Also, we theoretically prove that F' is the tightest upper bound if only the values of PSL, SSL, SL are used in the computation. Namely, we have the following properties.

Table V. The Procedure for Computing $F'(2, 2, \mathsf{PSL}, \mathsf{SSL}, \mathsf{SL})$

S_{id}	SL	PSL	SSL	Step (1)	Step (2)	Step (3)
S_1	6	1	4	✓	(1+2)/6	
S_2	4	1	2	✓	(1+2)/4	✓
S_3	3	1	1			
S_4	3	1	1			
S_5	5	1	3	✓	(1+2)/5	✓

PROPERTY 6. *$F'(u, \mathsf{PSL}, \mathsf{SSL}, \mathsf{SL})$ is the tightest upper bound for arithmetic occupancy of any node in the subtree of P with u supporting sequences if we only use the values of PSL, SSL, SL to compute the bound.*

PROOF. We prove by contradiction. Assume that there exists another upper bound $\widetilde{F}$ s.t. $\widetilde{F}(u, \mathsf{PSL}, \mathsf{SSL}, \mathsf{SL}) < F'(u, \mathsf{PSL}, \mathsf{SSL}, \mathsf{SL})$ for any given parameters of u, PSL,SSL, SL. Then, we can construct a sequence database $\mathcal{D}_P$ s.t. $|PS_i| = \mathsf{PSL}[i]$, $|SS_i| = \mathsf{SSL}[i]$, $|S_i| = \mathsf{SL}[i]$ where PS_i and SS_i are the prefix sequence and suffix sequence of P in any S_i ($S_i \in \mathcal{D}_P$), respectively. Furthermore, there exists any pattern P' in the subtree rooted at P s.t. $|\mathcal{D}_{P'}| = u$ and $\phi(P') = F'(u, \mathsf{PSL}, \mathsf{SSL}, \mathsf{SL})$. If we apply the upper bound $\widetilde{F}$ to the data of $\mathcal{D}_P$, the occupancy of any pattern P' will be no bigger than $\widetilde{F}(u, \mathsf{PSL}, \mathsf{SSL}, \mathsf{SL})$. However, we have a pattern P' in this subtree s.t. $\phi(P') = F'(u, \mathsf{PSL}, \mathsf{SSL}, \mathsf{SL})$ and $F'(u, \mathsf{PSL}, \mathsf{SSL}, \mathsf{SL}) \le \widetilde{F}(u, \mathsf{PSL}, \mathsf{SSL}, \mathsf{SL})$, thus the contradiction occurs.

The construction of the database $\mathcal{D}_P$ is quite straightforward. Note that for any pattern P' with $|\mathcal{D}_{P'}| = u$ in the subtree of P,

$$F'(u, \mathsf{PSL}, \mathsf{SSL}, \mathsf{SL}) = \frac{1}{u} \cdot \max_{l_1,\ldots,l_u} \sum_{i=1}^{u} \frac{\mathsf{PSL}[l_i] + v}{\mathsf{SSL}[l_i]},$$

there exists $l_1^*, \ldots, l_u^*$ s.t.

$$\phi(P') = \frac{1}{u} \sum_{i=1}^{u} \frac{\mathsf{PSL}[l_i^*] + v}{\mathsf{SL}[l_i^*])}.$$

Then we can construct the database $\mathcal{D}_P = \{S_1, \ldots, S_n\}$ (n is the length of SL) s.t.

$$|PS_i| = \mathsf{PSL}[i], |SS_i| = v, |S_i| = \mathsf{SL}[i],$$

for $1 \le i \le n$, and $|P'| - |P| = v$. Then, the $\mathcal{D}_P$ can be constructed such that P' has no other supporting sequences except $S_{l_i^*}$. Then

$$\phi(P') = \frac{1}{u} \sum_{i=1}^{u} \frac{\mathsf{PSL}[l_i^*]) + v}{\mathsf{SL}[l_i^*]} = F'(u, \mathsf{PSL}, \mathsf{SSL}, \mathsf{SL}).$$

The procedure in Table V actually generates the supporting transactions, on which the occupancy is exactly equal to this upper bound. Therefore, $F'(u, \mathsf{PSL}, \mathsf{SSL}, \mathsf{SL})$ is the tightest one. □

PROPERTY 7. $\max_{freq_{min} \le u \le \mathcal{D}_P}(\frac{u}{|\mathcal{D}|} + \lambda \cdot F'(u, \mathsf{PSL}, \mathsf{SSL}, \mathsf{SL}))$ *is the tightest upper bound of the arithmetic quality (arithmetic occupancy used inside) for any node in the subtree of P if we only use the values of PSL, SSL, SL to compute the bound.*

PROOF. The proof is trivial based on Property 6. □

4.2.3. Tradeoff Between Tightness and Efficiency. So far, we have proposed two quality bounds. Although F' is provably tighter than F, when $\mathsf{SSL}^{\downarrow}[freq_{min}]$ is large, it also

Table VI. The Complexity of Different Estimation Methods

Quality upper bound	Complexity
F	$O(n \cdot log(n))$
F'	$O(n \cdot log(n) \cdot \mathsf{SSL}^{\downarrow}[freq_{min}])$
$\hat{F}$	$O(n \cdot log(n) \cdot m)$

takes much more time than F and possibly becomes the computing bottleneck. To alleviate this situation, we propose a new technique that achieves balance between the bound tightness and computational efficiency. The basic idea is as follows. Instead of enumerating every possible value of SSL in the range of $[0, \mathsf{SSL}^{\downarrow}[freq_{min}]]$, we split this large interval into m smaller intervals $[v_0, v_1 - 1], \ldots, [v_{m-1}, v_m - 1]$ ($v_0 = 0, v_m = \mathsf{SSL}^{\downarrow}[freq_{min}] + 1$).

Note that $v = |P'| - |P|$, for each interval $[v_{k-1}, v_k - 1]$, using the assumption that $v_{k-1} \leq v < v_k$, we obtain a tighter bound on occupancy (quality) in Property 8 (9). At last, we unify the m bounds (much smaller than $\mathsf{SSL}^{\downarrow}[freq_{min}]$) and get a final result. The time complexity for computing this quality bound is $O(n \cdot log(n) \cdot m)$. Using a proper value for m, we achieve the tradeoff between the efficiency and effectiveness of the bound. The complexity for these three methods are summarized in Table VI.

PROPERTY 8. *For any pattern P' in the subtree of P, let u be the frequency of P', $v = (|P'| - |P|)$. Assume $v_k \leq v < v_{k+1}$, then the occupancy of P' satisfies that*

$$\phi(P') \leq \frac{1}{u} \cdot \max_{\substack{l_1,\ldots,l_u, \\ SSL[l_i] \geq v_k}} \sum_{i=1}^{u} \frac{PSL[l_i] + min(SSL[l_i], v_{k+1} - 1)}{SL[l_i]}. \tag{29}$$

PROOF. The right-hand side of the previous equation is denoted as $\hat{F}(u, v_k, v_{k+1}, \mathsf{PSL}, \mathsf{SSL}, \mathsf{SL})$. The proof combines the ideas in the proof of Property 2 and the proof of Property 4. Since we have assumed $v_k \leq |P'| - |P| \leq v_{k+1} - 1$, combining this with $|\mathcal{D}_{P'}| = u$. Thus, for any pattern P' as described

$$\phi(P') = \frac{1}{u} \sum_{S \in \mathcal{D}_{P'}} \frac{|P'|}{|S|} \tag{30}$$

$$= \frac{1}{u} \sum_{S \in \mathcal{D}_{P'}} \frac{|P| + |P'| - |P|}{|S|} \tag{31}$$

$$\leq \frac{1}{u} \sum_{\substack{S \in \mathcal{D}_{P'}, \\ \mathsf{SSL}[l_S] \geq v_k}} \frac{\mathsf{PSL}[l_S] + min(\mathsf{SSL}[l_i], v_{k+1} - 1)}{\mathsf{SL}[l_S]} \tag{32}$$

$$\leq \frac{1}{u} \cdot \max_{\substack{l_1,\ldots,l_u, \\ \mathsf{SSL}[l_i] \geq v_k}} \sum_{i=1}^{u} \frac{\mathsf{PSL}[l_i] + min(\mathsf{SSL}[l_i], v_{k+1} - 1)}{\mathsf{SL}[l_i]}. \quad \square \tag{33}$$

PROPERTY 9. *Let $u' = freq_{min}$ be the minimum frequency threshold. For any integers $0 = v_0 < \cdots < v_m = SSL^{\downarrow}[u'] + 1$,*

$$\hat{F}(u', PSL, SSL, SL) = \max_{0 \leq k < m} \hat{F}(u', v_k, v_{k+1}, PSL, SSL, SL),$$

is the upper bound on arithmetic occupancy for any frequent P' in the subtree of P. And

$$\max_{u' \leq u \leq |\mathcal{D}_P|} u/|\mathcal{D}| + \lambda \cdot \hat{F}(u, PSL, SSL, SL)),$$

is the upper bound on quality for any P'.

Table VII. An Example of Sequence Database with Weights

S_{id}	Sequence	SW	PSW	SSW
S_1	a[5]**b[3]**c[2]d[3]e[2]g[2]	$\langle 5, 3, 2, 3, 2, 2\rangle$	$\langle 3\rangle$	$\langle 2, 3, 2, 2\rangle$
S_2	a[5]**b[6]**c[5]f[3]	$\langle 5, 6, 5, 3\rangle$	$\langle 6\rangle$	$\langle 5, 3\rangle$
S_3	a[6]**b[6]**c[5]	$\langle 6, 6, 5\rangle$	$\langle 6\rangle$	$\langle 5\rangle$
S_4	a[3]**b[4]**c[5]	$\langle 3, 4, 5\rangle$	$\langle 4\rangle$	$\langle 5\rangle$
S_5	a[4]**b[3]**e[2]h[2]i[3]	$\langle 4, 3, 2, 2, 3\rangle$	$\langle 3\rangle$	$\langle 2, 2, 3\rangle$

PROOF. The proof is trivial based on Property 8. □

4.3. Extension on Item Weights

So far, in this article all items in sequence databases are treated uniformly, but real items have different importance [Tao 2003]. Thus, we extend the definition of occupancy to the situation when the weights on the items are considered. A weight function $\mathcal{W}$ is used to store the map of weights to items in a sequence database. For example, Table VII shows a sequence database with the item weight, in which the number in the square bracket followed by any item is the weight of the item occurrence in a sequence. For example, $\mathcal{W}(a, S_1)$ is 5. The occupancy with weighted items is defined as follows.

Definition 4.4 (*Weighted Occupancy*). The *occupancy* of a pattern P with item weight is defined as

$$\phi(P) = AVG\left(\left\{\frac{\sum_{i\in P}\mathcal{W}(i, S)}{\sum_{j\in S}\mathcal{W}(j, S)} : S \in \mathcal{D}_P\right\}\right),$$

Obviously, the original occupancy is the special case when all item weights are equal to 1. For example, the weighted occupancy of pattern abc in Table VII is $\frac{1}{4}\cdot(\frac{10}{17}+\frac{16}{19}+\frac{17}{17}+\frac{12}{12})$. To compute the upper bound with item weight, we need the following notions w.r.t. P.

Definition 4.5 (*SW& PSW& SSW*). The *Sequence Weight matrix* of P, denoted as SW, is used to record the weight vector of each sequence S ($S \in \mathcal{D}_P$). The *Prefix Sequence Weight matrix* of P, denoted as PSW, is used to record the weight vector of each prefix sequence in S. The *Suffix Sequence Weight matrix* of P, denoted as SSW, is used to record the weight vector of each suffix sequence in S.

Note that SW, PSW, and SSW are three matrices, in which the first index refers to the sequence number and the second refers to the index of an item in a sequence. Table VII also shows SW, PSW, and SSW of pattern b. As shown in this Table, $\mathsf{SW}[5][1] = 4$, $\mathsf{PSW}[5][1] = 3$ and $\mathsf{SSW}[5][1] = 2$.

Similar to the arithmetic occupancy with no item weight in Section 4.2, we also propose two bounds with item weight in the same sprits. The first bound is more efficient, while the second one is the tightest upper bound with certain input constraints. Specifically, we give the following properties and their proofs are similar to that in Section 4.2.

PROPERTY 10 (THE EFFICIENT UPPER BOUND ON ϕ). *For any frequent pattern P' in the subtree of P*

$$\phi(P') \le \frac{1}{u'}\max_{l_1,\ldots,l_{u'}}\sum_{i=1}^{u'}\frac{\sum_{j=1}^{|\mathsf{PSW}[l_i]|}\mathsf{PSW}[l_i][j]+\sum_{j=1}^{|\mathsf{SSW}[l_i]|}\mathsf{SSW}[l_i][j]}{\sum_{j=1}^{|\mathsf{SW}[l_i]|}\mathsf{SW}[l_i][j]}, \tag{34}$$

where u' is the minimum frequency threshold.

Table VIII. An Example of Sequence Database with Multiple Itemsets

S_{id}	Sequence	PSL	SSL	SL
S_1	⟨**{a,b}**,{a,c,d},{c,e,g}⟩	2	6	8
S_2	⟨**{a,b}**,{b,c,f}⟩	2	3	5
S_3	⟨**{a,b}**,{a,c,d}⟩	2	3	5
S_4	⟨**{a,b}**,{c,d},{c,e,g}⟩	2	5	7
S_5	⟨**{a,b}**,{e,h},{h,i}⟩	2	4	6

PROPERTY 11 (THE "TIGHTEST" UPPER BOUND ON ϕ). *For any frequent pattern P' in the subtree of P*

$$\phi(P') \le \frac{1}{u'} \cdot \max_{0 \le v \le SSL^{\downarrow}[u']} F(u, v, PSW, SSW, SW), \tag{35}$$

and

$$F(u, v, PSW, SSW, SW) = \max_{\substack{l_1, l_2, \ldots, l_{u'} \\ SSL[l_i] \ge v}} \sum_{i=1}^{u'} \frac{\sum_{j=1}^{|PSW[l_i]|} PSW[l_i][j] + \sum_{j=1}^{v} SSW[l_i]^{\downarrow}[j]}{\sum_{j=1}^{|SW[l_i]|} SW[l_i][j]}, \tag{36}$$

where u' is the minimum frequency threshold, SSL can be directly obtained from SSW and $SSW[l_i]^{\downarrow}$ is a weight vector in descending order, for example, $SSW[1]^{\downarrow}$ in Table VII is $\langle 3, 2, 2, 2 \rangle$.

We can also prove that the second is the tightest bound on arithmetic occupancy using only PSW, SSW, SW. Similarly, we may achieve the tradeoff between the bound tightness and computational efficiency, and induce the corresponding upper bound on quality.

In Section 5.2, we will show how to set the item weights in the real application of print-area recommendation and how these item weights improve the performance.

4.4. Extension on Multiple-Itemset Sequences

Here, we discuss how the proposed algorithm DOFRA is extended to handle the multiple-itemset sequences. Table VIII gives an example database with multiple-itemset sequences.

First, we extend the occupancy definition for multiple-itemset sequences. Specifically,

Definition 4.6 (*Occupancy for multiple-itemset sequences*). The *occupancy* of a pattern P for multiple-itemset sequences defined as

$$\phi(P) = AVG\left(\left\{\frac{Num(P)}{Num(S)} : S \in \mathcal{D}_P\right\}\right),$$

Where $Num(P)$ means the number of items in P and $Num(S)$ means the number of items in S.

Take the database shown in Table VIII as an example. The occupancy (arithmetic) for pattern $\langle\{a, b\}\rangle$ is $\frac{1}{5} \cdot (\frac{2}{8} + \frac{2}{5} + \frac{2}{5} + \frac{2}{7} + \frac{2}{6}) \approx 0.33$. Note that the proposed techniques for computing upper bounds can be directly applied for this definition.

As [Ayres et al. 2002] shown that there are two kinds of extensions to grow a certain prefix sequence, namely, sequence-extension (S-extension) and itemset-extension (I-extension). A sequence extension w.r.t a prefix is generated by appending a new event consisting of single item to the prefix sequence, while an itemset extension w.r.t a prefix is generated by adding an item to any one event of the prefix sequence. Instead of using only S-extension in Figure 3 for frequent single-itemset sequences enumeration, we can use both S-extension and I-extension for frequent multiple-itemset sequences enumeration. Since we only need statistical information such as the length information (SL, PSL, and SSL) or weight information (SW, PSW, and SSW) for computing the upper bounds, we can apply the similar idea to compute the upper bounds for the scenario of I-extension.

5. EVALUATION ON EFFECTIVENESS

In this section, we evaluate the effectiveness of occupancy in two real applications, namely the print-area recommendation and the travel-landscape recommendation. Specifically, we demonstrate the following: (1) how the concept of occupancy help to improve the recommendation performance compared with the baseline, (2) how the weighted occupancy help to improve the recommendation performance, (3) how the different occupancy definitions affect the recommendation performance, and (4) how the occupancy weight λ affects the recommendation performance.

5.1. Experimental Setup

5.1.1. Motivated Applications and Datasets.

Print-area Recommendation. Assume that we have the log database which records how previous users clipped the Web pages from a Web site. Each transaction refers a set of content clips selected on a Web page. Given a Web page from the same Web site, we aim to recommend the informative clips for this Web page. More specifically, let $\mathcal{I}$ be the complete set of distinct clips in the database. For a given Web page we can get a set $Q \subseteq \mathcal{I}$ of clips which are included in this Web page. Thus, our task is to select a subset of Q for the clip recommendation. By mining top qualified itemset the recommended itemset is the one $F \subseteq Q$ which has the maximal quality value among the qualified itemsets. To show the effectiveness of the proposed method we manually labeled the ground-truth of print-areas on the 2,000 Web pages from the 100 major print-worthy Web sites[4].

Travel-landscape Recommendation. We exploit a real world travel data set used in Liu et al. [2011] for evaluating the effectiveness of occupancy on a sequence database. This data set is provided from a travel company in China. There are nearly 220,000 purchase records (packages) from tourists with the travel time from January 2000 to October 2010. Each record can be viewed as a sequence of landscapes. Note that each record has a location attribute. For a tourist who wants to have a journey in a city named “C”, we first collect all records whose location attribute equals to “C” as a sequence database. Based on this sequence database, we aim to recommend a sequence of interesting landscapes in ‘C’ with the maximal quality value to the given tourist. To show the effectiveness of the proposed method, we filter out the purchase records with unavailable landscapes and finally obtain $22,929$ records from 198 cities.

5.1.2. Evaluation Methods and Metrics. We compare the proposed solution with the maximal frequent itemset based method (introduced in Section 2.3). Specifically, we can first generate all maximal frequent itemsets and among them select the one with the largest number of items for recommendation.

In the application of print-area recommendation, for each Web page from a Web site, we use leave-one-out cross validation to evaluate the recommendation accuracy, that is, we iteratively select one page as query and the log data on the left Web pages from the same Web site are used to generate the transaction database[5] for recommendation. A recommendation result actually refers to a set of content clips on the given Web page.

[4]For details about this dataset, please refer to http://home.ustc.edu.cn/∼stone/DOFIA-Appendix.pdf.
[5]We transform this transaction database as a sequence database by lexically ordering the items, on which we can run DOFRA.

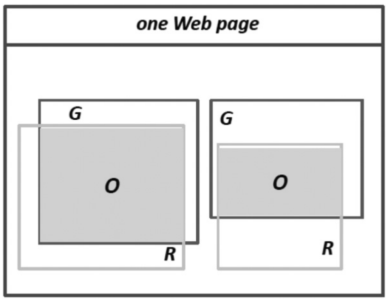


Fig. 4. The region overlap for a Web page.

Table IX. The Performance of the Baseline Method on the 2,000 Web Pages

α	$Pre(\%)$	$Rec(\%)$	$F1(\%)$
0.0	91.51	93.16	90.73
0.1	92.04	93.69	**91.85**
0.2	90.77	90.78	89.52
0.3	90.25	90.63	88.89
0.4	87.80	87.56	86.06
0.5	83.26	81.07	79.36

Then, we can average these performance values over the 2,000 Web pages to get the average performance. Similarly, in the application of travel-landscape recommendation, for each user's consumed record from a city, we iteratively select one record from a city as query and the log data on the left records from the same city are used to generate the sequence database for recommendation. Then, we can average these performance values over the 22,929 records to get the average performance.

For the purpose of evaluation, we can calculate the precision Pre, recall Rec, and $F1$ score between the recommendation pattern and the ground truth query. Specifically

$$Pre = \frac{I(R, G)}{I(R)}, Rec = \frac{I(G, R)}{I(G)}, F_1 = 2 \times \frac{Pre \times Rec}{Pre + Rec}, \tag{37}$$

For different applications, $I(.,.)$ has different meanings. In the print-area recommendation, since a recommendation result refers to a set of content clips on the given Web page, we can evaluate its effectiveness by calculating the overlap area between the recommended clips (R) and the ground truth on the query page (G). Thus, $I(R, G)$ in print-area recommendation is the overlap area between R and G and $I(R)$ (or $I(G)$) means the area of R (or G). As shown in Figure 4, the two blue rectangles refer to the two clips selected by a user while the two yellow ones represent the two clips recommended by DOFRA, and the area O is the overlap area. In travel-landscape recommendation, a recommended pattern actually refers to a sequence of landscapes in the given city. Since the items in a pattern have the sequential order, thus, $I(R, G)$ is the size of the longest common subsequences (LCS) [Gorbenko 2012] between R and G and $I(R)$ (or $I(G)$) is the number of items in R (or G).

5.2. Results on Print-Area Recommendation

The results of the baseline method are summarized in Table IX. The results of DOFRA are shown in Table X. Note that for the baseline method, the maximal F_1 score is 91.85% ($\sigma = 0.18$) when $\alpha = 0.1$, while DOFRA with arithmetic occupancy and item weights achieves a maximal F_1 score of 93.68% ($\sigma = 0.15$) when $\lambda = 0.1$. So the improvement on recommendation accuracy is clear. We also conduct significance tests on the difference

Table X. The Performance of DOFRA on the 2,000 Web Pages ($\alpha = 0.05$ and $\beta = 0.5$)

	Harmonic occupancy						Arithmetic occupancy					
	Without item weights			With item weights			Without item weights			With item weights		
λ	*Pre*(%)	*Rec*(%)	$F1$(%)	*Pre*(%)	*Rec*(%)	$F1$(%)	*Pre*(%)	*Rec*(%)	$F1$(%)	*Pre*(%)	*Rec*(%)	$F1$(%)
0.0	92.68	93.35	91.77	92.71	93.77	92.03	92.71	93.35	91.79	92.86	94.14	92.40
0.05	91.98	93.03	91.27	92.94	95.40	93.27	92.22	93.19	91.50	93.00	95.57	93.40
0.1	92.55	94.13	92.34	93.03	95.70	**93.48***	92.55	94.12	92.33	93.17	95.91	**93.68***
0.2	93.05	94.63	92.81	92.85	95.45	93.24	92.96	94.99	**93.02***	92.84	95.63	93.33
0.3	92.87	94.78	**92.84***	92.62	95.36	93.07	92.43	94.56	92.54	92.72	95.35	93.10
0.4	92.84	94.57	92.55	92.36	95.11	92.76	92.37	94.37	92.26	92.46	95.22	92.88
0.5	92.67	94.54	92.44	92.1	94.77	92.44	92.17	94.36	92.15	92.20	94.81	92.51
1.0	91.61	92.9	90.95	91.29	94.00	91.42	91.44	92.78	90.83	91.39	94.08	91.51
$+\infty$	85.67	85.83	83.09	90.48	93.09	90.21	85.96	85.22	82.55	90.63	92.96	90.16
Average	91.77	93.04	91.10	92.32	94.69	92.43	91.65	92.96	91.10	92.42	94.81	92.56

*Passes the significance test (significantly larger than the baseline $\alpha = 0.1$) at the confidence level of 0.05.

of different methods with the best results. All tests use the t-statistic and set the confidence level to 0.05.

Since we have considered the area size of content clips in our evaluation, it is natural to set the weight of a clip to its area size on the Web page. Intuitively, if a content clip covers a large area of a Web page, we should give it more weight in computing occupancy. Thus, it is more likely to be recommended. We look at the entries in Table X to see the effects of item weight for different occupancy definitions. The average $F1$ score of the models with item weights is 1.33% higher than the models without item weights for harmonic occupancy, so does arithmetic occupancy. We can also observe that the model with arithmetic occupancy is slightly better than that with harmonic occupancy in this dataset. Note that best results of DOFRA significantly larger than the baseline with $\alpha = 0.1$ at the confidence level of 0.05.

The role of λ in DOFRA to this application is quite interesting. Intuitively, λ represents the emphasis we put on occupancy. A higher emphasis on occupancy is expected to lead to a better recall. Such is indeed the case—when λ increases from 0.0 to around 0.2, recommendation recall increases smoothly for the models. Interestingly, recommendation precision increases at the same time. The main reason is that with the increase of λ the quality value (considering both support and occupancy) may lead to better patterns whose area intersection with the ground truth have both better precision and better recall. However, when λ is too large, the performance of DOFRA deteriorates. Too much emphasis on occupancy tends to find patterns with a very small support just above the threshold α and the recommendation quality will naturally drop greatly. In practice, λ can be chosen by cross validation to achieve the best performance in similar manners. Note that the line of $\lambda = 0$ presents the results by using closed frequent pattern mining algorithm (support is only considered). We can observe that DOFRA outperforms the closed frequent pattern mining method.

5.3. Results on Travel Recommendation

The results of the baseline method and DOFRA are shown in Tables XI and XII, respectively. Note that for the baseline method in Table XI, the maximal F_1 score is 87.11% (when $\alpha = 0.4$), while DOFRA achieves a maximal F_1 score of 87.99% (when $\lambda = 0.8$) for harmonic occupancy. We can also find that the best performance with harmonic occupancy and that with arithmetic occupancy is very close. The analysis of λ in DOFRA to this application is similar to the print-area application which have been discussed in Section 5.2. Note that best results of DOFRA do not significantly larger than the baseline with $\alpha = 0.4$ at the confidence level of 0.05. The reason may be

Table XI. The Performance of the Baseline Method on the 22,929 Travel Package Purchase Records

α	$Pre(\%)$	$Rec(\%)$	$F1(\%)$
0.0	54.00	76.54	60.87
0.1	70.67	87.32	75.74
0.2	72.79	88.28	77.45
0.3	88.67	86.00	86.00
0.4	90.40	86.51	**87.11**
0.5	91.20	85.52	86.78

Table XII. The Performance of DOFRA on the 22,929 Travel Package Purchase Records ($\alpha = 0.01$ and $\beta = 0.5$)

	Harmonic occupancy			Arithmetic occupancy		
λ	$Pre(\%)$	$Rec(\%)$	$F1(\%)$	$Pre(\%)$	$Rec(\%)$	$F1(\%)$
0.0	94.36	68.49	77.64	94.46	68.28	77.53
0.2	93.88	79.45	84.40	93.99	79.33	84.38
0.4	93.73	81.09	85.39	93.79	80.41	84.97
0.6	92.92	82.76	86.06	92.93	82.71	86.04
0.8	91.68	86.73	**87.99**	92.87	82.94	86.14
1.0	90.42	86.93	87.46	90.86	86.88	**87.69**
1.2	90.27	87.01	87.41	90.36	87.00	87.47
1.6	90.04	87.15	87.36	90.23	87.20	87.51
2.0	90.00	87.19	87.37	89.93	87.06	87.25
$+\infty$	86.93	86.12	84.99	86.93	86.12	84.99

that the records in the travel-package data are less diversified than that in print-area dataset.

In addition, we also present the top qualified pattern from "Beijing" for a case study, that is, ⟨*Tian An Men, Jingshan Park, Temple of Heaven, The Lama Temp, Forbidden City, Beihai Park, Summer Palace, The Great Wall*⟩. Note that the first six landscapes are near together in the center of Beijing city (in the second ring road) while *Summer Palace* is far away from the center of the city (in fifth ring road) and *The Great Wall* is out of "Beijing". Indeed, those landscapes in this pattern are ordered in spatial order. Moreover, this pattern covers almost all of the famous landscapes in Beijing.

One may think that it is hard to judge the recommended route for "Beijing" is a good one without considering the time-constraint and moneytary cost. Without such constraints, it is intuitive that more places in the recommendation, the better the recommendation should be. However, if we know each user's time and money constraints, it is easy to extend our method for pattern recommendation to satisfy each user's needs. There are two possible extensions. The first is to obtain the sequential database by filtering the log data that do not satisfying user's time and money constraints. Based on this filtered database, then we can directly run DOFRA for high quality pattern recommendation. This method is called *pre-process* one. Another alternative method is called *in-process* one. To be specific, when running DOFRA, the current searched pattern is qualified if satisfying not only support and occupancy constraints but also time and money constraints.

6. EVALUATION ON EFFICIENCY

In this section we present the empirical evaluation on the efficiency of the proposed algorithm DOFRA over the real and large synthetic data sets. Specifically, we compare DOFRA's running time with the baseline method to our problem. Here, the baseline method is to find all frequent closed patterns first, then compute each pattern's occupancy and quality and output top qualified one among them. We implemented

Table XIII. Characteristic of Datasets

Datasets	#Seq.	#Items	AveLen	MaxLen	Type
Gazelle	29,369	1,423	3	651	Very sparse
Msnbc	989,818	17	5	14,795	Sparse
Snake	175	20	67	121	Dense
D100kC15.	100,000	3,000	15	41	Little dense

Table XIV. Default Parameters of Datasets

Datasets	α	β	λ	k
Gazelle	0.0002	0.5	1	1
Msnbc	0.0003	0.5	1	1
D100kC15D3k	0.001	0.5	1	1
Snake	0.7	0.5	1	1

BIDE [Wang and Han 2004] as the baseline because of its high efficiency to find frequent closed patterns. In addition, we also implemented DOFRA_W (using the bound techniques proposed in Section 4.3) and BIDE_W to mine the qualified patterns on the weighted data. Note that a nontop-k version of DOFRA (i.e., $k = 1$) is used for the comparison with BIDE.

6.1. Datasets and Experimental Environment

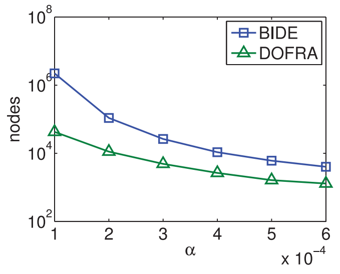
In Section 5, we have used two datasets: 2,000 Web pages from 100 Web sites and 22,929 travel packages from 198 cities. For effectiveness evaluation, we iteratively select one Web page (or travel) as query and the log data on the left Web pages (or travel packages) from the same Web site (or city) are used to generate the transaction (or sequence) database. We can find that the average size for each Web site or each city is very small, it is not appropriate to use these two datasets for efficiency evaluation. In order to better evaluate the efficiency of DOFRA, we adopt three real datasets and one synthetic datset with different characteristics in our experiments.

The first real dataset is *Gazelle* [Kohavi et al. 2000] and is very sparse, which contains totally 29,369 customer's Web click-stream data. The second real dataset is *Msnbc* and is sparse. It contains totally 989,818 customer's Web click-stream data[6]. The third real datasets is *Snake* and is dense. It contains totally 175 Toxin-Snake protein sequences and 20 unique items. The last dataset is a generated dataset *D100kC15N3k* by IBM synthetic data generator [Agrawal and Srikant 1995], which is little dense. The detailed characteristics of the four datasets are summarized in Table XIII.

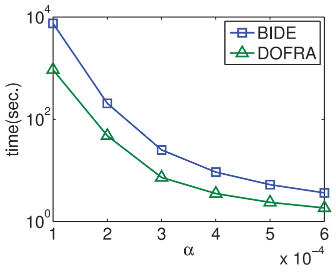
We will check the performance changes with different settings of problem parameters (i.e., α, β, λ and k). The default parameters for each dataset are summarized in Table XIV (the default setting of α follows a principle: the denser the dataset, the larger α). We will adjust one parameter while fixing all the others to see the efficiency changes. Then, we will compare the tradeoff between bound tightness and computational efficiency. Finally, we will show the effectiveness of the upper bounds of weighted occupancy proposed in Section 4.3. In this article, we only present the performance on α over the four data sets. For the description succinct, we then show the performance on β, λ, k, the replication factor, the tradeoff parameter and the comparison of DOFRA_W and BIDE_W over the data *Gazelle* (similar results can also be found when considering the other three datasets). By default, we use the upper bounds proposed in Section 4.2.1.

All the algorithms were implemented in C++ and experiments were performed on a Windows 7 laptop with quad core Intel i5-2450M processor and 4GB of main memory.

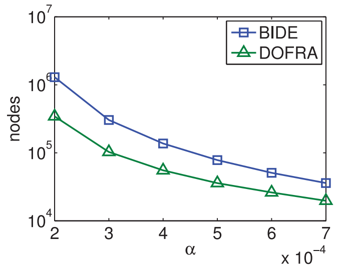
[6]downloaded from http://archive.ics.uci.edu/ml/datasets/.



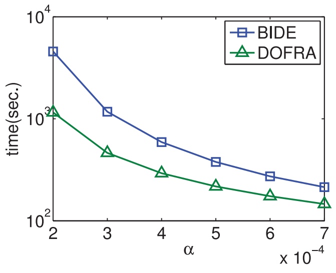
(a) *Gazelle* versus nodes



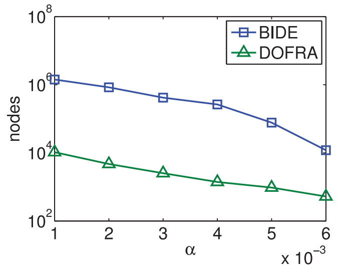
(b) *Gazelle* versus time



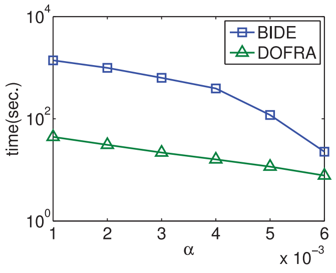
(c) *Msnbc* versus nodes



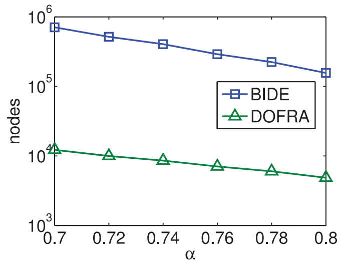
(d) *Msnbc* versus time



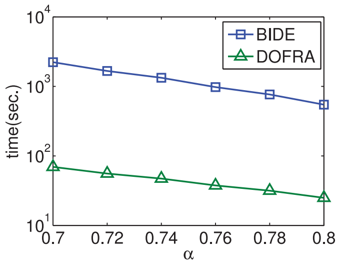
(e) *T100kC15D3k* versus nodes



(f) *T100kC15D3k* versus time



(g) *Snake* versus nodes



(h) *Snake* versus time

Fig. 5. Comparison of DOFRA and BIDE w.r.t. α over the four datasets.

6.2. Evaluation on Different Problem Settings

The support threshold. In practice, it is often desirable to have a low frequency threshold in order to detect more diverse patterns, which leads to the problem of too many frequent patterns and long running time. Figure 5(a) and Figure 5(b) demonstrate the number of searched nodes and running time between DOFRA and BIDE on dataset *Gazelle* for support thresholds varying from 0.0001 to 0.0006. We can see that DOFRA always searches less nodes and runs much faster than BIDE. For example, at

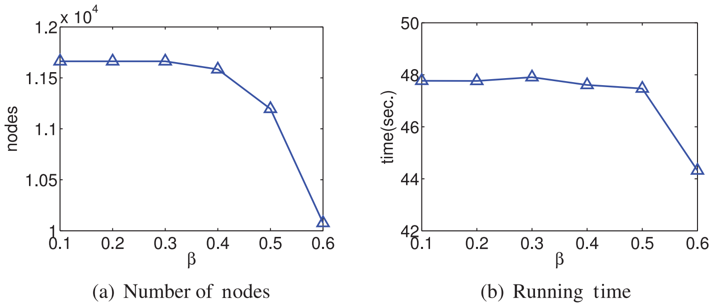


Fig. 6. The effects of β on DOFRA over *Gazelle*.

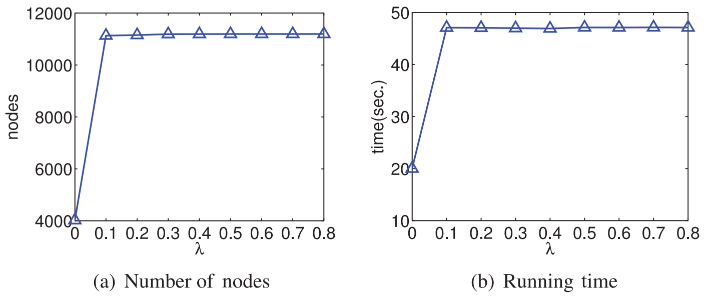


Fig. 7. The effects of λ on DOFRA over *Gazelle*.

support 0.0001, DOFRA is more than 8 times faster than BIDE while it only searches about 1.9% nodes of those searched by BIDE. Similar results can also be found in Figures 5(c) through 5(h) for other datasets. Depending on different characteristics of the datasets, the effect of pruning can be different. Specifically, the effect of pruning is increasing when the dataset becomes dense. For example, for the very dense dataset *Snake*, DOFRA runs about 32 times faster than BIDE when α is lower than 0.7. This indicates that our method can work well especially on very dense databases which makes other baselines prohibitive.

The occupancy threshold. As shown in Figure 6, with the increase of the occupancy threshold β, the running time and the number of the visited nodes decreases moderately for DOFRA. For example, as β increases from 0.1 to 0.6, the number of nodes searched decreases from $11,662$ to $10,076$ and the running time decreases slightly from 47.7s to 44.3s. One might expect that the increase in running time is exponential, but since we are doing the top qualified search, the most qualified node that DOFRA has encountered helps a lot in reducing the search space, even if we set a low value in occupancy threshold. This observation gives us a lot of flexibility in choosing a proper value for β.

The occupancy weight parameter. The occupancy weight parameter λ reflects the priority we put on occupancy. As can be seen from Figure 7, with the increase of λ from 0 to 0.1, the number of nodes visited by DOFRA increases from $4,021$ to $11,194$ and the running time increases from $20s$ to $47s$. When $\lambda > 0.1$, the number of nodes visited and running time converges because occupancy is already playing the major role here, and thus the increase on λ has little effect on the behavior of the algorithm.

Top *k*. In practice, we often want to mine more than just the top few qualified patterns for recommendation. With more patterns we can even do ranking on these patterns and find a set of diverse and high-quality patterns. Here, we increase k to check the performance of DOFRA. As can be seen from Figure 8, both the number of searched

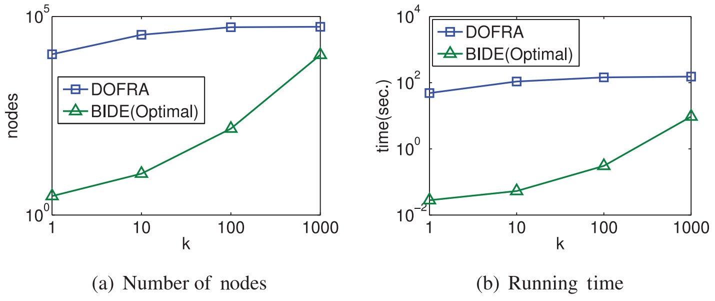


Fig. 8. The effects of top-k on DOFRA.

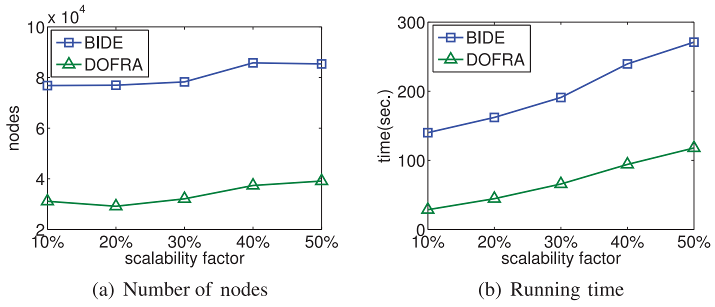


Fig. 9. The effects of scalability factor on BIDE and DOFRA over *Msnbc*.

nodes and the running time grow linearly from $48.5s$ to $152.7s$ with the increase of k from 1 to 1000. Similar to Wu et al. [2012], in order to show the tradeoff performance, we also compare the proposed top-k algorithm DOFRA with BIDE tuned with the optimal threshold to discover the same amount of pattern (denoted as BIDE(Optimal)). As can be observed from Figure 8, the gap between the two algorithms becomes smaller as k increases.

6.3. Evaluation on the Scalability Factor

Here, we test the scalability of the algorithm on different sizes of databases. Specifically, we use the $981,898$ sequences from *Msnbc* as the basic database and then scale it up by using 10%, 20%, 30%, 40%, and 50% of this database. The results are shown in Figure 9, including the running time and the number of nodes in prefix projected database tree searched by BIDE and DOFRA over *Msnbc* with the scalability factor from 10% to 50%. As can be seen from Figure 9(a), DOFRA only searches about 40% nodes of those searched by BIDE. With respect to running time, DOFRA's running time grows linearly, from $28.6s$ for 10% of *Msnbc* to $117.9s$ for 50% of *Msnbc*. BIDE's running time also grows linearly from $140s$ for 10% of *Msnbc* to $271s$ for 50% of *Msnbc*, but DOFRA is much faster than BIDE.

6.4. Tradeoff Between Tightness and Efficiency

In Section 4.2.3, we proposed a technique that achieves tradeoff between bound tightness and computational efficiency. Specifically, instead of enumerating each possible value of v in the range of $[0, \mathsf{SSL}^{\downarrow}[freq_{min}]]$ as F' does, we suggest to split the large interval into smaller intervals $[v_0, v_1 - 1], \ldots, [v_{K-1}, v_K - 1]$, $(v_0 = 0, v_K = \mathsf{SSL}^{\downarrow}[freq_{min}] + 1)$. Figure 10 shows the effect of this technique on the data set *Gazelle* with $\alpha = 0.0005$, $\beta = 0.5$, $\lambda = 0.001$. The x-axis shows the "interval length", that is, $v_i - v_{i-1}$ (we divide the interval evenly) and the y-axis shows the number of nodes searched and the running

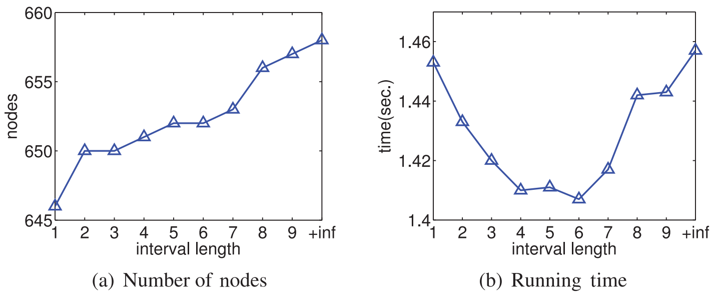


Fig. 10. Tradeoff between tightness and efficiency over *Gazelle*.

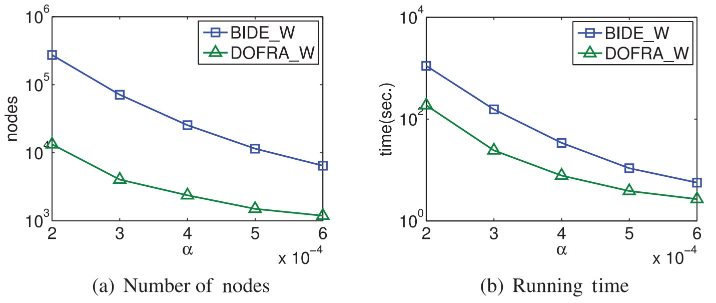


Fig. 11. Comparison of DOFRA_W and BIDE_W w.r.t. α over *Gazelle* with weights.

time in the two figures respectively. We can see that smaller the interval length, tighter the bound and fewer the nodes searched. However, the shortest running time does not occur when the bound is tightest (646 nodes searched when interval length=1), since the computation is too costly. And also the shortest time does not occur when the efficient bound is used (658 nodes searched when interval length=+inf). In fact, the overall performance is the tradeoff between bound tightness and computational efficiency. In this case, the algorithm is fastest when the interval length is 6.

6.5. The Effectiveness of the Weighted Occupancy Upper Bounds

In Section 4.3, we extend occupancy to weighted occupancy and propose the corresponding efficient and tightest upper bounds for weighted occupancy. To show the effectiveness of the proposed upper bounds, we extend DOFRA and BIDE to DOFRA_W and BIDE_W to mine the qualified patterns on weighted database. In order to get the corresponding weighted sequential databases for *Gazelle*, we take a similar method in Tseng et al. [2013], Wu et al. [2012], and Liu and Qu [2012] and generate the weights for items randomly ranging from 1 to 100. Figure 11(a) and 11(b) demonstrate the number of searched nodes and running time between DOFRA_W and BIDE_W on dataset *Gazelle* (with weights) for support thresholds varying from 0.0002 to 0.0006.We can see that DOFRA_W always searches less nodes and runs much faster than BIDE_W. For example, at support 0.0002, DOFRA_W is more than 6 times faster than BIDE_W while it only searches about 4.9% nodes of those searched by BIDE_W.

7. RELATED WORK

Frequent pattern mining [Agrawal et al. 1993] is a fundamental research problem in data mining with many broad applications, such as association-rule-based classification [She et al. 2003] and clustering [Aggarwal et al. 2007]. There is a great amount of work that studies efficient mining of frequent patterns [Pasquier et al. 1999; Han et al.

2000a, 2000b; Burdick et al. 2001; Zaki 2001; Ayres et al. 2002; Gouda and Zaki 2005; Lu et al. 2011; Kam et al. 2013; Cai et al. 2014; Chen and Chen 2014; Fournier-Viger et al. 2014; Lam et al. 2014; Wu et al. 2014] and we refer to [Han et al. 2007] as a recent survey on this field. These algorithms can be classified into mining frequent patterns [Han et al. 2000a; Pei et al. 2001], frequent closed patterns [Pasquier et al. 1999; Wang and Han 2004] and frequent maximal patterns [Burdick et al. 2001; Luo and Chung 2005]. To reduce the number of frequent patterns, some interestingness measures and constraints are proposed [Mannila and Toivonen 1997; Garofalakis et al. 1999; Pei et al. 2001; Bonchi and Lucchese 2004; Gade et al. 2004; Xiong et al. 2006; Pei et al. 2007; Li et al. 2012] along with algorithms incorporating them for efficient pattern mining.

The concept of *occupancy* proposed in this article can be viewed as a new interestingness measure and constraint. It can be seen as the *relative size* of a pattern to its supporting transactions (sequences) rather than the *absolute size* of it, and might be more meaningful in many applications. With the concept of occupancy, we then formulate the problem of mining top-*k* qualified patterns which maximize the weighted sum of support and occupancy.

One work very similar to the spirits of ours is [Wang et al. 2005], in which Wang et al. formulated the problem of finding the top-*k* frequent patterns with their sizes no smaller than a threshold. The main difference between our work and theirs can be summarized into two facets. First, Wang et al. [2005] used the absolute size of a pattern as a constraint, while we use the measure occupancy, namely the relative size of a pattern to its supporting transactions (sequences). As argued before, in many cases occupancy might be a more reasonable constraint. In addition, since occupancy is a normalized value, it also makes the tuning easier. Second, Wang et al. [2005] tried to find the top-*k* frequent closed pattern, so frequency was used as the quality measure. In our work, we define quality as the weighted sum of support and occupancy, which might be more appropriate.

Similar to Geerts et al. [2004] and Gade et al. [2004], we also use a branch-and-bound algorithm by exploiting anti-monotonic constraints to reduce the search space. However, Geerts et al. [2004] and Gade et al. [2004] proposed to maximize the *area or tile* (product of pattern size and its frequency) while the proposed occupancy is used to measure the completeness of a pattern in its supporting sequences (or transactions). The detailed upper bounds for area and occupancy are very different. Recent works [Cerf et al. 2009; Soulet and Crémilleux 2009; Soulet et al. 2011] have also proposed methods to deal with complex constraints constructed from primitives. In those works, the primitives are required to be monotonic/anti-monotonic/convex, none of which is a property of occupancy. But it might be an interesting work to extend these frameworks to handle occupancy properly.

Some works formulated constraints [Bonchi and Lucchese 2007; Guns et al. 2011] in a more general framework. To be specific, Bonchi and Lucchese [2007] proposed a general and effective framework named $ExAMiner^{GEN}$ by exploiting the properties of constraints with *anti-monotonic*, *monotonic*, *succinct*, *convertible*, and *loose anti-monotonic*, comprehensively. [Guns et al. 2011] used constraint programming for solving itemset mining problems in a declarative way and they incorporated some well-known constraints like *monotonic*, *anti-monotonic*, *convertible* in the constraint programming system. Different from many other constraints, occupancy is not *anti-monotonic*, *monotonic*, *succinct*, *convertible*, and *loose anti-monotonic*, thus, the previous general frameworks cannot be leveraged for our problems.

8. CONCLUDING REMARKS

Motivated by some real-world pattern recommendation applications, in this article we introduced a new interesting pattern measure *occupancy* to measure the completeness of a pattern in its supporting transactions or sequences. Based on the definition of occupancy, we then formulated the problem of mining top-*k* qualified patterns. To tackle this problem, we explored the upper bound properties on occupancy and propose an efficient algorithm DOFRA that injects these properties deeply into the search process. Specifically, we proposed two upper bounds for *arithmetic occupancy*, in which the first one is more efficient and the second one is theoretically proved to be tightest with certain input constraints. Also, we showed the technique that achieves the balance between the two upper bounds. Moreover, we also showed that these techniques can be applied when we consider the weights of items in calculating the pattern occupancy. In addition, DOFRA can be further enhanced by exploiting the properties of closed patterns since the top qualified pattern must be closed. Finally, we showed the effectiveness of occupancy in the two real-world applications, namely the print-area recommendation for Web pages and the travel-landscape recommendation. And, we also systematically evaluated DOFRA on real and synthetic data sets and the results demonstrated that DOFRA significantly outperforms the baseline method in terms of efficiency.

In the future, we would like to extend occupancy and exploit some novel applications based on *multiple-itemset* sequences. Note that the commonality of the proposed applications about print-area recommendation and travel-landscape recommendation is that the items in each transaction or sequence work as a whole for a task. Thus, it may be interesting to apply occupancy to some traditional frequent pattern mining applications [Han et al. 2007] (e.g., drug design, genome analysis, market basket analysis, and click stream analysis), in which the items in each transaction or sequence work as a whole for a task. In addition, it may be interesting to exploit occupancy-based frequent patterns for pattern-based classification [She et al. 2003] and clustering [Aggarwal et al. 2007].

REFERENCES

Charu C. Aggarwal, Na Ta, Jianyong Wang, Jianhua Feng, and Mohammed Zaki. 2007. Xproj: A framework for projected structural clustering of xml documents. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining.* ACM, 46–55.

Rakesh Agrawal, Tomasz Imielinski, and Arun Swami. 1993. Mining association rules between sets of items in large databases. In *Proceedings of the International Conference on Management of Data.* ACM, Washington, DC, USA, 207–216.

Rakesh Agrawal and Ramakrishnan Srikant. 1995. Mining sequential patterns. In *Proceedings of the International Conference on Data Engineering.* IEEE, 3–14.

Jay Ayres, Jason Flannick, Johannes Gehrke, and Tomi Yiu. 2002. Sequential pattern mining using a bitmap representation. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining.* ACM, 429–435.

Francesco Bonchi and Claudio Lucchese. 2004. On closed constrained frequent pattern mining. In *Proceedings of the International Conference on Data Mining*. IEEE, 35–42.

Francesco Bonchi and Claudio Lucchese. 2007. Extending the state-of-the-art of constraint-based pattern discovery. *Data Knowl. Eng.* 60, 2, 377–399.

Doug Burdick, Manuel Calimlim, and Johannes Gehrke. 2001. MAFIA: A maximal frequent itemset algorithm for transactional databases. In *Proceedings of the International Conference on Data Engineering.* IEEE, 443–452.

Guochen Cai, Chihiro Hio, Luke Bermingham, Kyungmi Lee, and Ickjai Lee. 2014. Sequential pattern mining of geo-tagged photos with an arbitrary regions-of-interest detection method. *Expert Systems With Applications* 41, 7, 3514–3527.

Loïc Cerf, Jérémy Besson, Céline Robardet, and Jean-François Boulicaut. 2009. Closed patterns meet n-ary relations. *ACM Trans. Knowl. Discov. Data* 3, 1, 3:1–3:36.

Jingyu Chen and Ping Chen. 2014. Sequential pattern mining for uncertain data streams using sequential sketch. *Journal of Networks* 9, 2, 252–259.

Philippe Fournier-Viger, Cheng Wei Wu, Antonio Gomariz Penalver, and Vincent S. Tseng. 2014. VMSP: Efficient vertical mining of maximal sequential patterns. In *Proceedings of the 27th Canadian Conference on Artificial Intelligence (AI 2014).* Springer International Publishing, 83–94.

Krishna Gade, Jianyong Wang, and George Karypis. 2004. Efficient closed pattern mining in the presence of tough block constraints. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.* ACM, Seattle, WA, USA, 138–147.

Minos N. Garofalakis, Rajeev Rastogi, and Kyuseok Shim. 1999. SPIRIT: Sequential pattern mining with regular expression constraints. In *Proceedings of the 25th International Conference on Very Large Data Bases.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA. 223–234.

Floris Geerts, Bart Goethals, and Taneli Mielikainen. 2004. Tiling databases. In *Proceedings of the 7th International Conference Discovery Science.* Springer Berlin Heidelberg, 278–289.

Anna Gorbenko. 2012. On the longest common subsequence problem. *Applied Mathematical Sciences* 6, 116, 5781–5787.

Karam Gouda and Mohammed J. Zaki. 2005. Genmax: An efficient algorithm for mining maximal frequent itemsets. *Data Mining and Knowledge Discovery* 11, 3, 223–242.

Tias Guns, Siegfried Nijssen, and Luc De Raedt. 2011. Itemset mining: A constraint programming perspective. *Artif. Intell.* 175, 12–13, 1951–1983.

L. R. Haff. 1979. An identity for the Wishart distribution with applications. *Journal of Multivariate Analysis* 9, 4, 531–544.

Jiawei Han, Hong Cheng, Dong Xin, and Xifeng Yan. 2007. Frequent pattern mining: Current status and future directions. *Data Mining and Knowledge Discovery* 15, 1, 55–86.

Jiawei Han, Pei Jian, and Yiwen Yin. 2000a. Mining frequent patterns without candidate generation. In *Proceedings of the International Conference on Management of Data*. ACM, Dallas, Texas, USA, 1–12.

Jiawei Han, Jian Pei, Behzad Mortazavi-Asl, Qiming Chen, Umeshwar Dayal, and Meichun Hsu. 2000b. FreeSpan: Frequent pattern-projected sequential pattern mining. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining*. ACM, Boston, Massachusetts, USA, 355–359.

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

C. L. Kam, C. Raissi, M. Kaytoue, and J. Pei. 2013. Mining statistically significant sequential patterns. In *Proceedings of the International Conference on Data Mining*. IEEE, 488–497.

Ron Kohavi, Carla E. Brodley, Brian Frasca, Llew Mason, and Zijian Zheng. 2000. KDD-Cup 2000 organizers' report: Peeling the onion. *SIGKDD Explor. Newsl.* 2, 2, 86–93.

Hoang Thanh Lam, Fabianand Morchen, Dmitriy Fradkin, and Toon Calders. 2014. Mining compressing sequential patterns. *Statistical Analysis and Data Mining* 7, 1, 34–53.

Chun Li, Qingyan Yang, Jianyong Wang, and Ming Li. 2012. Efficient mining of gap constrained subsequences and its various applications. *ACM Trans. Knowl. Discov. Data* 6, 1, 1–39.

Mengchi Liu and Junfeng Qu. 2012. Mining high utility itemsets without candidate generation. In *CIKM*. ACM, 55–64.

Qi Liu, Yong Ge, Zhongmou Li, Enhong Chen, and Hui Xiong. 2011. Personalized travel package recommendation. In *Proceedings of the International Conference on Data Mining.* IEEE, 407–416.

Eric Hsueh-Chan Lu, Vincent S. Tseng, and Philip S. Yu. 2011. Mining cluster-based temporal mobile sequential patterns in location-based service environments. *IEEE Transactions on Knowledge and Data Engineering* 23, 6, 914–927.

Congnan Luo and Soon Myoung Chung. 2005. Efficient mining of maximal sequential patterns using multiple samples. In *Proceedings of the SIAM International Conference on Data Mining*. SIAM, 415–426.

Heikki Mannila and Hannu Toivonen. 1997. Levelwise search and borders of theories in knowledge discovery. *Data Mining and Knowledge Discovery* 1, 3, 241–258.

Nicolas Pasquier, Yves Bastide, Rafik Taouil, and Lotfi Lakhal. 1999. Discovering frequent closed itemsets for association rules. In *Proceedings of the 7th International Conference on Data Theory.* Springer Berlin Heidelberg, 398–416.

Jian Pei, Jiawei Han, and LVS Lakshmanan. 2001. Mining frequent itemsets with convertible constraints. In *Proceedings of the International Conference on Data Engineering*. IEEE, 433–442.

Jian Pei, Jiawei Han, Behzad Mortazavi-asl, Helen Pinto, Qiming Chen, Umeshwar Dayal, and Mei chun Hsu. 2001. PrefixSpan: Mining sequential patterns efficiently by prefix-projected pattern growth. In *Proceedings of the International Conference on Data Engineering*. IEEE, 215–224.

Jian Pei, Jiawei Han, and Wei Wang. 2007. Constraint-based sequential pattern mining: The pattern-growth methods. *J. Intell. Inf. Syst.* 28, 2, 133–160.

Rong She, Fei Chen, Ke Wang, Martin Ester, Jennifer L. Gardy, and Fiona S. L. Brinkman. 2003. Frequent-subsequence-based prediction of outer membrane proteins. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining* ACM, Washington, DC, 436–445.

Arnaud Soulet and Bruno Crémilleux. 2009. Mining constraint-based patterns using automatic relaxation. *Intell. Data Anal.* 13, 1, 109–133.

Arnaud Soulet, Chedy Raïssi, Marc Plantevit, and Bruno Cremilleux. 2011. Mining dominant patterns in the sky. In *Proceedings of the International Conference on Data Mining*. IEEE, 655–664.

Linpeng Tang, Lei Zhang, Ping Luo, and Min Wang. 2012. Incorporating occupancy into frequent pattern mining for high quality pattern recommendation. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*. ACM, Maui, Hawaii, USA, 75–84.

Feng Tao. 2003. Weighted association rule mining using weighted support and significant framework. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining*. ACM, Washington, DC, 661–666.

V. S. Tseng, Bai-En Shie, Cheng-Wei Wu, and P. S. Yu. 2013. Efficient algorithms for mining high utility itemsets from transactional databases. *IEEE Tran. on Know. and Data Engi.* 25, 8, 1772–1786.

Jianyong Wang and Jiawei Han. 2004. BIDE: Efficient mining of frequent closed sequences. In *Proceedings of the International Conference on Data Engineering*. IEEE, 79–90.

Jianyong Wang, Jiawei Han, Ying Lu, and P. Tzvetkov. 2005. TFP: An efficient algorithm for mining top-k frequent closed itemsets. *IEEE Tran. Know. Data Engi.* 17, 5, 65–663.

Cheng Wei Wu, Bai-En Shie, Vincent S. Tseng, and Philip S. Yu. 2012. Mining top-K high utility itemsets. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining*. ACM, 78–86.

Youxi Wu, Lingling Wang, Jiadong Ren, Wei Ding, and Xindong Wu. 2014. Mining sequential patterns with periodic wildcard gaps. *Applied Intelligence* 41, 1, 99–117.

Hui Xiong, Pang-Ning Tan, and Vipin Kumar. 2006. Hyperclique pattern discovery. *Data Mining and Knowledge Discovery* 13, 2, 219–242.

Mohammed J. Zaki. 2001. SPADE: An efficient algorithm for mining frequent sequences. *Machine Learning* 42, 1–2, 31–60.

Received January 2014; revised November 2014; accepted March 2015