Incorporating Occupancy into Frequent Pattern Mining for High Quality Pattern Recommendation

Linpeng Tang[13*], Lei Zhang[23*], Ping Luo[3] and Min Wang[3]
[1] Shanghai Jiao Tong University ,[2] University of Science and Technology of China ,[3] HP Labs China
{Linpeng.tang, lei.zhang7, ping.luo, min.wang6}@hp.com

ABSTRACT

Mining interesting patterns from transaction databases has attracted a lot of research interest for more than a decade. Most of those studies use *frequency*, the number of times a pattern appears in a transaction database, as the key measure for pattern interestingness. In this paper, we introduce a new measure of pattern interestingness, *occupancy*. The measure of occupancy is motivated by some real-world pattern recommendation applications which require that any interesting pattern X should occupy a large portion of the transactions it appears in. Namely, for any supporting transaction t of pattern X, the number of items in X should be close to the total number of items in t. In these pattern recommendation applications, patterns with higher occupancy may lead to higher recall while patterns with higher frequency lead to higher precision. With the definition of occupancy we call a pattern *dominant* if its occupancy is above a user-specified threshold. Then, our task is to identify the *qualified* patterns which are both frequent and dominant. Additionally, we also formulate the problem of *mining top-k qualified patterns*: finding the qualified patterns with the top-k values of any function (e.g. weighted sum of both occupancy and support).

The challenge to these tasks is that the *monotone* or *anti-monotone* property does not hold on occupancy. In other words, the value of occupancy does not increase or decrease monotonically when we add more items to a given itemset. Thus, we propose an algorithm called DOFIA (DOminant and Frequent Itemset mining Algorithm), which explores the upper bound properties on occupancy to reduce the search process. The tradeoff between bound tightness and computational complexity is also systematically addressed. Finally, we show the effectiveness of DOFIA in a real-world application on print-area recommendation for Web pages, and also demonstrate the efficiency of DOFIA on several large synthetic data sets.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications—*Data Mining*

*This work was done when Linpeng and Lei were visiting HP Labs China.

CIKM'12, October 29–November 2, 2012, Maui, HI, USA.

Keywords

Frequent pattern mining, Constraint-based mining, Frequent and Dominant pattern, DOFIA

1. INTRODUCTION

Frequent pattern mining has been attracting abundant research interest for over a decade. Most of those studies use *frequency*, namely the number of times a pattern appears in a transaction database, as the key measure for pattern interestingness. In this paper we introduce a new measure, *occupancy*, which considers the degree that a pattern occupies the items in its supporting transactions. Specifically, we prefer the pattern that occupies a large portion of the transactions it appears in. This new measure of occupancy is motivated by some real-world applications of pattern recommendation and we describe two of the such applications below.

The first application is on print-area recommendation for Web pages. We often find that the printout generated by a Web browser's print function is far from satisfactory since it usually contains a large portion of irrelevant content (e.g., navigation menu, advertisements, and related links). To solve this problem, HP provides *Smart Print*[1] that contains a user-friendly interface so that a user can easily select her interested print areas. Such selections are stored in print logs with user consent. If we view each content clip (a selected content area) as an item and all the selected clips by a user on a given Web page as a transaction of items, the print log data from all users form a transaction database. Our task is to recommend an itemset (i.e., a set of content clips) in a given Web page to users based on this database. Naturally, the itemset we recommend should occur frequently to reflect the interests of most users. Equally important is the completeness of the itemset: it should occupy a large portion of the transactions it appears in so that the user would not feel that the recommendation is missing too much relevant content.

Another motivating application is on investment portfolio recommendation. Assume that we have a transaction database that contains a large set of *high-quality and diversified* investment portfolios. Each transaction represents the set of financial assets (e.g., stocks, bonds,funds etc.) owned by an experienced investor. Our goal is to mine the "interesting patterns" (i.e., high-quality and diversified patterns) from the database to recommend to new investors. Similarly, we prefer the investment patterns that appear frequently in the database. More importantly, a good investment portfolio usually works as an entirety to achieve investment balance and reduce risks. So we would expect that a good investment pattern should cover a large portion of the transactions in which it appears. For example, we have two patterns X, Y of equal fre-

[1]A Web browser extension, www.hp.com/go/smartprint

quency. If X covers 90% assets of its supporting transactions while Y only covers 30%, it is natural to consider X as a much better investment pattern. Thus, the occupancy of a pattern is very critical to this application.

The commonality of the above two applications is that the set of items in each transaction *works as an entirety for a task* and thus making occupancy as important a factor as frequency for recommending interesting patterns. In these applications, occupancy becomes another pattern interestingness measure which is an indispensable complement to frequency (or support): we consider a pattern interesting if it is not only frequent, but also as complete as possible in its supporting transactions. Intuitively, the support of a recommended pattern correlates to the recommendation precision while its occupancy is related to the recommendation recall.

We show that the value of occupancy does not increase or decrease monotonically when we add more items to a given itemset. As a result, previous techniques for pruning the search space for frequent itemset mining based on the anti-monotonicity of support do not apply in our setting. We explore the properties for the upper bounds of occupancy and quality (the weighted sum of support and occupancy) for various patterns and use the upper bound to prune the search process for high-efficient qualified pattern mining. The contributions of our work can be summarized as follows:

• We introduce a new interestingness measure *occupancy* for pattern mining problems. It is an indispensable complement to the widely-studied measure of support when the items in a transaction serve as a whole for certain tasks. We formulate the problem of mining top-k qualified patterns which maximize any increasing function of support and occupancy. In this study the weighted sum of support and occupancy is adopted as a show case.

• We propose an algorithm called DOFIA (DOminant and Frequent Itemset mining Algorithm) to solve the problem of qualified pattern mining. In DOFIA, we explore the properties on the upper bound of occupancy for the patterns to prune the search process for high-efficient pattern mining. Specifically, we propose two upper bounds on occupancy for any given itemset X and its supersets in the search tree. The first bound is computationally efficient, while the second is proved to be the tightest bound with certain input constraint. We also show techniques to achieve tradeoff between these two bounds.

• We demonstrate the importance of applying occupancy measure in pattern recommendation applications and the effectiveness of DOFIA through a real-world application of print-area recommendation for Web pages using real data. We show the introduction of occupancy results in significant improvement on the quality of the recommendation and the high performance of DOFIA in finding the top quality patterns.

• We systematically evaluate the efficiency of DOFIA on some large synthetic data sets. Specifically, we evaluate the impact on the efficiency of the algorithm by varying the transaction database (i.e., the number of transactions, the average length of transactions, etc.), the parameter settings (i.e., α, β, etc.), and the different strategies used in DOFIA.

The rest of the paper is organized as follows. We first present the formulation of the problem in Section 2. In Section 3 we give the overview of the proposed algorithm DOFIA. The details of DOFIA, especially on how to calculate the upper bounds on occupancy and quality, are discussed in Section 4. We report the empirical study to show the effectiveness and efficiency of DOFIA in Sections 5 and 6. We present the related works in Section 7 and conclude the paper in Section 8.

2. PRELIMINARIES AND PROBLEM FORMULATION

In this section we first give some preliminaries in pattern mining and propose the definitions of occupancy. Then, we formulate the qualified pattern mining problems which consider both support and occupancy.

2.1 Definitions and Notations

A transaction database is a set of transactions, where each *transaction* is a set of items. Let $\mathcal{I}$ be the complete set of distinct items and $\mathcal{T}$ be the complete set of transactions. Any non-empty set of items is called an *itemset* and any set of transactions is called a *transaction set*. The transactions that contain all the items in an itemset X are the *supporting transactions* of X, denoted as $\mathcal{T}_X$. The *frequency* of an itemset X (denoted as $freq(X)$) is the number of transactions in $\mathcal{T}_X$. The definitions of support and frequent itemset are adopted from [1].

In the motivating applications in Section 1, the itemset we intend to find should occupy a large portion of the transactions in which it appears. We can calculate the occupancy degree as follows. For an itemset X we identify all its supporting transactions $\mathcal{T}_X$. For each transaction $t \in \mathcal{T}_X$ we calculate the ratio of $\frac{|X|}{|t|}$. We then aggregate these ratios to compute a single value of occupancy for X. In this paper we focus on the average of these ratios while other aggregate functions such as *quantile* or *min* may also be considered. The definition of occupancy is given as follows.

Definition 1 (Occupancy): Formally, the *occupancy* of an itemset X is defined as

$$\phi(X) = average(\{\frac{|X|}{|t|} : t \in \mathcal{T}_X\}),$$

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

Then, we can use two different average functions, *harmonic average* and *arithmetic average*, in the above definition. Due to space limitations, we only consider the harmonic average in this paper, but similar techniques can also be employed when considering arithmetic average [2].

Definition 2 (Harmonic occupancy): The harmonic average of a set of numbers A is $HAverage(A) = \frac{|A|}{\sum_{a \in A} \frac{1}{a}}$. The *harmonic occupancy* of an itemset X is defined as

$$\phi_H(X) = HAverage(\{\frac{|X|}{|t|} : t \in \mathcal{T}_X\}) = \frac{|\mathcal{T}_X||X|}{\sum_{t \in \mathcal{T}_X} |t|}$$ ■

It is clear that the occupancy of an itemset X is the average ratio of the occurrences of the items in X to the number of the items in the transaction it appears in. The high value of the occupancy indicates that besides the items in X there are only a small number of items left inside the supporting transactions of X. Let us take the transaction database in Table 1[3] as an example. We calculate the harmonic occupancy of the two itemsets $I_1 = \{2, 7, 14, 20\}$ and $I_2 = \{2, 7, 14, 15, 20\}$ as follows. The supporting transactions of I_1 are $\{t_1, t_2, t_3, t_4, t_7, t_8, t_9\}$ while the supporting transactions of I_2 are $\{t_1, t_2, t_4\}$.

$$\phi_H(I_1) = \frac{4 \times 7}{12 + 8 + 5 + 11 + 4 \times 3} \approx 0.54$$

$$\phi_H(I_2) = \frac{5 \times 3}{12 + 8 + 11} \approx 0.48$$

[2]The comparison on effectiveness between harmonic and arithmetic average is out of the scope of this paper.

[3]This is a real world example taken from the HP's print log database.

Table 1: The transaction database of example 1

Trans No.	Length	Items
t_1	12	1 2 4 7 8 9 10 14 15 16 20 21
t_2	8	2 5 7 9 12 14 15 20
t_3	5	2 7 13 14 20
t_4	11	1 2 4 5 7 8 14 15 18 20 21
t_5	6	2 3 7 11 14 21
t_6	12	1 2 5 6 7 9 12 14 15 17 19 21
t_7	4	2 7 14 20
t_8	4	2 7 14 20
t_9	4	2 7 14 20
t_{10}	3	2 14 20

One may think that the itemset containing more items leads to a bigger value of occupancy. However, it is not always true. Consider itemsets I_1 and I_2. Even though $I_1 \subsetneq I_2$, we have $\phi_H(I_1) > \phi_H(I_2)$. The reason is that I_2 only appears in large transactions where it only occupies a small fraction, while I_1 appears in many smaller transactions where it occupies a large fraction. Thus, occupancy does not always increase monotonically when we add more items to an itemset. Similarly, we can show that occupancy does not always decrease monotonically when we add more items to an itemset either. This non-monotonic property of occupancy is in contrast to that of support in frequent pattern mining.

Definition 3 (Dominant Itemset): For a given minimum occupancy threshold β ($0 < \beta \leq 1$), X is said to be *dominant* if $\phi(X) \geq \beta$.■

With the definition of support and occupancy we can measure the *quality* of an itemset by combining these two factors.

Definition 4 (Quality): The *quality* of an itemset X is defined as $q(X) = \sigma(X) + \lambda\phi(X)$, where $\sigma(\cdot)$ denotes the (relative) support of a transaction, occupancy weight $0 \leq \lambda < +\infty$ is a user defined parameter to capture the relative importance of support and occupancy. ■

Here we use the weighted sum of support and occupancy as the quality. But it should be noted that any other functions, such as the harmonic average (similar to the F1 score), the sum of logarithms (similar to block size proposed in [8]) can be used to combine the two values of support and occupancy. Additionally, the proposed algorithms in this paper are independent of this function.

Definition 5 (Qualified Itemset): For a given minimum support threshold α and a minimum occupancy threshold β ($0 < \alpha, \beta \leq 1$), X is said to be *qualified* if $\sigma(X) \geq \alpha$ and $\phi(X) \geq \beta$. ■

2.2 Problem Formulation

Some pattern recommendation applications require that interesting patterns should be both frequent and dominant. On one hand, if a pattern X is frequent it means that there are enough cases such a pattern appears in the transaction database. Thus, it improves the recommendation precision. On the other hand, if X is dominant it indicates that the recommendation of X is complete enough. Thus, it guarantees the recommendation recall. Therefore, with the definition of support and occupancy, we formulate the problem of mining qualified patterns as follows.

Mining Qualified Patterns. This task is to find all the qualified patterns (which are both frequent and dominant) in a transaction database for the given support threshold α and the occupancy threshold β.

Additionally, 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.

Table 2: The transaction database of example 2

Trans No.	Items
t_1	b c
t_2	a b
t_3	a b c
t_4	a b d
t_5	a b c d

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

$$\arg\max_{X:\sigma(X)\geq\alpha,\phi(X)\geq\beta} (\sigma(X) + \lambda\phi(X)) \quad (1)$$

Note that there may be multiple top qualified patterns if there are ties in the maximum quality values.

There are three parameters in the definition of the top qualified pattern, namely α, β, λ. If there is no itemset that is both frequent and dominant with respect to α, β, 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. Parameter λ, the occupancy weight, is a user defined parameter to capture the relative importance of support and occupancy.

In the rest of the paper, we will primarily focus on mining top (i.e., $k = 1$) qualified pattern first. We then show that the solution to top qualified pattern mining can be easily extended to solve the problem of mining 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 Itemset* mining may be adopted for finding top-k qualified patterns. (An itemset X is a maximal frequent itemset if X is frequent and no superset of X is frequent [6]). Given a support threshold we can get multiple maximal frequent itemsets, among which 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 itemsets, the number of items in a pattern is used as a measure in pattern selection. Compared with the concept of occupancy, this is actually the *absolute* size of an itemset while occupancy is the *relative* size of an itemset to the number of items in its supporting transactions. Thus, it is possible that the maximal frequent itemsets selected by such methods have very low occupancy, even lower than the minimal occupancy threshold, thus leading to a low recall in recommendation. For example, when $\alpha = 0.3$, $I_2 = \{2, 7, 14, 15, 20\}$ is a maximal frequent itemset for the transaction database in Table 1. However, its occupancy is much lower than its subset $I_1 = \{2, 7, 14, 20\}$ as described earlier. Secondly, 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 itemsets. The experimental results in Section 5 further validate our analysis here.

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

3. OVERVIEW OF DOFIA

The straightforward solution to the pattern mining problem is to first generate all the frequent itemsets, calculate the occupancy value for each frequent itemset, and then select the qualified itemset with maximum quality value. In this section we will show how

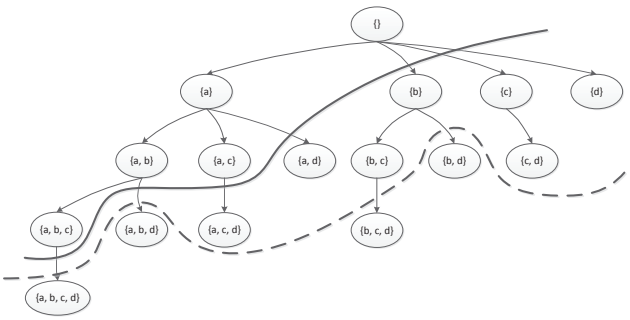


Figure 1: The lexicographic subset tree for the transaction database in Table 2. The solid line shows the search space of DOFIA, while the dashed line shows the search space for frequent pattern mining. $\alpha = 0.4, \beta = 0.3, \lambda = 1$**.**

the properties on occupancy and quality measures can be injected deeply into the search process and greatly prune the search space.

Pattern mining algorithms usually adopt the *lexicographic subset tree* to guide the search process. See Figure 1 as an example for four items a, b, c, d. The top element in the tree is the empty set and each lower level l contains all the l-itemsets (itemsets with exactly l items). The l-itemsets are ordered lexicographically on each level. Generating children in this manner enumerates all the distinct itemsets to be considered without redundancy. As there are 4 items, there are in total $2^4 = 16$ itemsets for consideration. Thus, in the lexicographic subset tree there are 16 nodes, each of which corresponds to an itemset.

Frequent pattern mining usually leverages the *monotonic decreasing* property of support values when adding more items to a given itemset. That is, if an itemset X is not frequent then all the supersets of X are not frequent either. Thus, the traversal in the tree is to find a cut (the dashed line in Figure 1) such that all the nodes (itemsets) above the cut are frequent, and all the nodes below the line are infrequent.

In this section we will show how to explore the properties on occupancy and quality to further prune the search space in the search tree. Specifically, given the root of a subtree we can estimate the upper bounds of the occupancy and quality values for all the nodes in this subtree. In other words, the occupancy and quality of any node in the given subtree will be no bigger than its upper bounds, respectively. If the upper bound on the occupancy is smaller than the 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 upper bounds less than $q*$ should be pruned.

Take the subtree with the root $\{b\}$ in Figure 1 as an example. We can give an upper bound of the quality for all the nodes in the subtree, including $\{b\}, \{bc\}, \{bd\}, \{bcd\}$. Assume we have already found $\{ac\}$ is the node with the highest quality 1.467 so far, which is bigger than the upper bound for the subtree rooted at $\{b\}$. Thus, subtree $\{b\}$ can be pruned. The solid line (which is above the dashed one) in Figure 1 is the cut line for pruning the search space for mining top qualified pattern. In this example, the search space when using only the frequency constraint for pruning has 12 nodes while the search space of DOFIA has only 5 nodes. The search space is greatly reduced by pruning using the upper bounds for occupancy and quality.

In Figure 1 the children of a node in the subset tree are ordered lexicographically on each level. It is worth mentioning that the order of these children also affects the algorithm efficiency. Ideally, we prefer to visit the nodes with the bigger quality values as early as possible since the big quality value can be used to prune the nodes whose quality upper bounds are less than it. On the other hand, we hope that the proposed upper bounds for occupancy and quality are as close as possible to the ground-truth values for a given subtree, which also helps to prune the search space. We consider two kinds of orders for the children of a node, i.e. the ascending and descending order of itemset support. Some experiments show that usually the ascending support order is more efficient than the other one (at least on the transaction databases used in the experiments). It seems the "fail first principle" applies in this case. The analysis on this observation is omitted.

Algorithm 1: DOFIA_DFS

input : the current Node $currentNode$, the top qualified pattern so far $bestNode$

```
1 if currentNode.quality > bestNode.quality then
2     bestNode ← currentNode;
3 for node ∈ currentNode.children do
4     supp ← node.support;
5     occu ← the occupancy bound on the subtree rooted at node;
6     qual ← the quality bound on the subtree rooted at node;
7     if (supp ≥ α) ∧ (occu ≥ β) ∧ (qual ≥ bestNode.quality)
      then
8         DOFIA_DFS(node, bestNode);
```

Algorithm 1 is the pattern mining process of a Depth-First-Search (DFS) with the pruning techniques. Note that since each node in the search tree uniquely corresponds to an itemset, we use the terms "node" and "itemset" interchangeably here. At $currentNode$ we first compare its quality with the top qualified node found so far. Then, we check any child, denoted by $node$, of $currentNode$. If $node$ is not frequent, or its occupancy upper bound is smaller than β, or its quality upper bound is smaller than the current maximal quality value, then $node$ is pruned. Otherwise, we recursively check $node$.

So far we have omitted the most challenging part in the algorithm: how to compute the upper bounds on occupancy and quality. Since the pruning of the search process is at the cost of computing these upper bounds, they should be computed efficiently. We describe its details in the next section.

4. THE UPPER BOUNDS OF OCCUPANCY AND QUALITY

In this section we will show how to efficiently compute the upper bounds of the harmonic occupancy and its quality. First, We give an overview of these upper bounds in Section 4.1.

4.1 Overview of the Upper Bounds

We give the notations which will be widely used in this study as follows. For any subtree, let X be the itemset for the subtree root (the root itemset), Y be the itemset including all the new items which will be extended in all the descendants of this subtree (the extension itemset). For example, for the subtree root $X = \{b\}$ in Figure 1 there are two new items c, d which appear in the descendants. Thus, $Y = \{cd\}$. With the extension itemset Y, we can get the following two vectors EL (Extension Length vector) and TL (Transaction Length vector) such that $\mathsf{EL}(i) = |t_i \cap Y|$ and $\mathsf{TL}(i) = |t_i|$, where t_i is any transaction in $\mathcal{T}_X$. For example, for the root set $X = \{ab\}$ with extension set $Y = \{cd\}$, the supporting transactions are $\mathcal{T}_X = \langle t_2, t_3, t_4, t_5 \rangle$, and thus $\mathsf{EL} = \langle |t_2 \cap Y|, |t_3 \cap Y|, |t_4 \cap Y|, |t_5 \cap Y| \rangle = \langle 0, 1, 1, 2 \rangle$, and $\mathsf{TL} = \langle |t_2|, |t_3|, |t_4|, |t_5| \rangle = \langle 2, 3, 3, 4 \rangle$. Furthermore, we will use u to denote the frequency of an itemset W in the subtree rooted at X,

and v to denote how many items are in W aside from the items in X, i.e. $v = |W - X|$. These notations are summarized in Table 3.

Furthermore, we will frequently need to sort EL, TL. For a vector V (V is EL or TL in our case), $\mathsf{V}^{\uparrow}$ ($\mathsf{V}^{\downarrow}$) is the vector obtained by sorting V in ascending (descending) order. So $\mathsf{EL}^{\downarrow}(1)$ will be the largest value in EL, and $\mathsf{TL}^{\uparrow}(i)$ will be the i-th smallest value in TL.

Table 3: Notations used by Section 3

X	The itemset for the root of a subtree
Y	The extension itemset for the corresponding subtree
W	Any itemset in the subtree X, $X \subseteq W \subseteq (X \cup Y)$
u	u often denotes $\lvert\mathcal{T}_W\rvert$
v	v often denotes $\lvert W - X\rvert$
EL	The Extension Length vector, $\mathsf{EL}(i) = \lvert t_i \cap Y\rvert$ where $t_i \in \mathcal{T}_X$
TL	The Transaction Length vector, $\mathsf{TL}(i) = \lvert t_i\rvert$ where $t_i \in \mathcal{T}_X$
$\mathsf{V}^{\uparrow}, \mathsf{V}^{\downarrow}$	vector V sorted by ascending/descending order

We aim to estimate the occupancy and quality upper bounds of all the nodes in the subtree of X. In other words, the occupancy (quality) of any node in the subtree of X will be less than this occupancy (quality) upper bound. The basic idea is briefly described as follows.

First, we assume that we know the frequency u of any itemset $W \subseteq (X \cup Y)$ in the subtree of X. Then, we will propose $F(u, |X|, \mathsf{EL}, \mathsf{TL})$ such that

$$\phi(W) \leq F(u, |X|, \mathsf{EL}, \mathsf{TL}), \tag{2}$$

It is worth mentioning that the computation of F only involves $u, |X|, \mathsf{EL}, \mathsf{TL}$. Using more detailed information in $\mathcal{T}_X$ would surely lead to better bounds, but it would also make the computation more costly. Also note that here F is dependent on u, the frequency of W.

Then, we will show $F(u, |X|, \mathsf{EL}, \mathsf{TL})$ is not increasing with the increase of u, namely

$$F(u+1, |X|, \mathsf{EL}, \mathsf{TL}) \leq F(u, |X|, \mathsf{EL}, \mathsf{TL}) \tag{3}$$

We call Equation 3 the *anti-monotonicity* property of the occupancy upper bound with respect to u.

Since it is required that W be frequent, its frequency u should not be less than the minimum frequency threshold $freq_{min}$, the minimal integer which is not smaller than $\alpha \cdot |\mathcal{T}|$. Thus, we have the following theorem.

Theorem 1 (Occupancy Upper Bound): For any itemset W in the subtree of X

$$\phi(W) \leq F(freq_{min}, |X|, \mathsf{EL}, \mathsf{TL}), \tag{4}$$

Proof: The conclusion holds if $F(u, |X|, \mathsf{EL}, \mathsf{TL})$ satisfies Inequalities (2) and (3). ∎

Note that in Theorem 1 it is not required to know the frequency of W. Thus, this is the occupancy upper bound of any node in the subtree of X. Next, for the upper bound of quality we also have

Theorem 2 (Quality Upper Bound): For any itemset W in the subtree of X

$$q(W) \leq \max_{freq_{min} \leq u' \leq |\mathcal{T}_X|} \left(\frac{u'}{|\mathcal{T}|} + \lambda F(u', |X|, \mathsf{EL}, \mathsf{TL})\right) \tag{5}$$

Proof: For any W in the subtree of X with the frequency u,

$$q(W) = \frac{u}{|\mathcal{T}|} + \phi(W) \leq \frac{u}{|\mathcal{T}|} + \lambda F(u, |X|, \mathsf{EL}, \mathsf{TL}), \tag{6}$$

Then, the conclusion holds if $F(u, |X|, \mathsf{EL}, \mathsf{TL})$ satisfies Inequality (2). ∎

Theorem 2 gives the quality upper bound of any node in the subtree of X. It should be noted that the result is independent of the function to combine support and occupancy. Any other functions can be used here and the quality upper bounds can be computed in similar ways.

In the following we will propose the F functions which satisfy Inequalities (2) and (3) for the harmonic occupancy.

4.2 The Upper Bounds of the Harmonic Occupancy

For harmonic occupancy 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 $|X|, \mathsf{EL}, \mathsf{TL}$ 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 $F(u, |X|, \mathsf{EL}, \mathsf{TL})$

We propose $F(u, |X|, \mathsf{EL}, \mathsf{TL})$ function as follows.

$$F(u, |X|, \mathsf{EL}, \mathsf{TL}) = \frac{u|X| + u\mathsf{EL}^{\downarrow}(u)}{\sum_{i=1}^{u} \mathsf{TL}^{\uparrow}(i)}, \tag{7}$$

Next, We propose Properties 1 and 2 to show that the F function in Equation (7) satisfies Inequalities (2) and (3).

Property 1: For any itemset W in the subtree of X, let u be the frequency of W. Then, the occupancy of W satisfies that

$$\phi_H(W) \leq F(u, |X|, \mathsf{EL}, \mathsf{TL}) \tag{8}$$

Proof: Let's consider the harmonic occupancy of W.

$$\phi_H(W) = \frac{u|W|}{\sum_{t \in \mathcal{T}_W} |t|} \tag{9}$$

$$= \frac{u|X| + u|W - X|}{\sum_{t \in \mathcal{T}_W} |t|} \tag{10}$$

$$\leq \frac{u|X| + u\mathsf{EL}^{\downarrow}(u)}{\sum_{t \in \mathcal{T}_W} |t|} \tag{11}$$

$$\leq \frac{u|X| + u\mathsf{EL}^{\downarrow}(u)}{\sum_{i=1}^{u} \mathsf{TL}^{\uparrow}(i)} \tag{12}$$

Note that Inequality (11) is due to the fact that since $|\mathcal{T}_W| = u$, there are at least u transactions whose extension lengths are not smaller than $|W - X|$, and so $|W - X|$ can be no larger than the smallest of these u lengths. ∎

Property 2: $F(u+1, |X|, \mathsf{EL}, \mathsf{TL}) \leq F(u, |X|, \mathsf{EL}, \mathsf{TL})$. ∎

Proof:

$$F(u+1, |X|, \mathsf{EL}, \mathsf{TL}) \tag{13}$$

$$= \frac{(u+1)|X| + (u+1)\mathsf{EL}^{\downarrow}(u+1)}{\sum_{i=1}^{u+1} \mathsf{TL}^{\uparrow}(i)} \tag{14}$$

$$\leq \frac{u(|X| + \mathsf{EL}^{\downarrow}(u)) + (|X| + \mathsf{EL}^{\downarrow}(u))}{(\sum_{i=1}^{u} \mathsf{TL}^{\uparrow}(i)) + \mathsf{TL}^{\uparrow}(u+1)} \tag{15}$$

$$\leq \frac{u|X| + u\mathsf{EL}^{\downarrow}(u)}{\sum_{i=1}^{u} \mathsf{TL}^{\uparrow}(i)} \tag{16}$$

$$= F(u, |X|, \mathsf{EL}, \mathsf{TL}) \tag{17}$$

Inequality (15) comes from $\mathsf{EL}^{\downarrow}(u+1) \leq \mathsf{EL}^{\downarrow}(u)$.Inequality (16) comes from the property that for any $a_1, a_2 \geq 0$ and $b_1, b_2 > 0$, $\frac{a_1}{b_1} \geq \frac{a_2}{b_2} \Rightarrow \frac{a_1}{b_1} \geq \frac{a_1+a_2}{b_1+b_2}$. It is easy to check that $\frac{u(|X|+\mathsf{EL}^{\downarrow}(u))}{\sum_{i=1}^{u} \mathsf{TL}^{\uparrow}(i)} \geq \frac{|X|+\mathsf{EL}^{\downarrow}(u)}{\mathsf{TL}^{\uparrow}(u+1)}$, so Inequality (16) follows.

With Properties 1 and 2 we can also easily prove that the results in Theorems 1 and 2 hold for $F(u, |X|, \mathsf{EL}, \mathsf{TL})$. Then, Algorithm 2 gives the pseudo code for computing the quality upper bound in Theorem 2 with $F(u, |X|, \mathsf{EL}, \mathsf{TL})$.

The complexity of Algorithm 2 is $O(n + \mathsf{TL}^{\downarrow}(1))$. We first use the time of $O(n + \mathsf{TL}^{\downarrow}(1))$ for sorting EL, TL with *counting sort algorithm*, and then $O(n)$ time for the loop in Line 5.

Note that if any other function is used in the *quality* definition to combine support and occupancy we can substitute the equation of $u/|\mathcal{T}| + \lambda \cdot occu$ in Line 9 with the used function to get the right answer.

Algorithm 2: The quality upper bound based on $F(u, |X|, \mathsf{EL}, \mathsf{TL})$

input : the root set X, the extension set Y, the corresponding two vectors EL, TL and $n = |\mathcal{T}_X|$.
output: $qual$, the quality upper bound of any itemset in the subtree rooted at X.

```
1 sort EL by descending order;
2 sort TL by ascending order;
3 sum ← 0;
4 qual ← −∞;
5 for u ← 1 to n do
6     sum ← sum + TL(u);
7     if u ≥ freq_min then
8         occu ← (u|X| + u · EL(u))/sum;
9         qual ← max(qual, u/|T| + λ · occu);
```

4.2.2 The "Tightest" Upper Bound $F'(u, v, |X|, \mathsf{EL}, \mathsf{TL})$

As mentioned before, $F(u, |X|, \mathsf{EL}, \mathsf{TL})$ has the parameter u, frequency for any itemset W in the subtree of X. Using a new parameter $v = |W \cap Y| = |W - X|$, i.e., the number of items in W aside from those in X, we may obtain a stronger upper bound.

Property 3: Let

$$F'(u, v, |X|, \mathsf{EL}, \mathsf{TL}) = \frac{u|X| + u \cdot v}{\min_{l_1, \cdots, l_u, \mathsf{EL}(l_i) \geq v} \sum_{i=1}^{u} \mathsf{TL}^{\uparrow}(l_i)} \tag{18}$$

Then for any itemset W in the subtree of X with $|\mathcal{T}_W| = u$ and $|W \cap Y| = v$, we have $\phi_H(W) \leq F'(u, v, |X|, \mathsf{EL}, \mathsf{TL})$.

Furthermore, for any itemset W' in the subtree of X with $|\mathcal{T}_{W'}| = u$ (and no constraint on $|W' \cap Y|$), we have

$$\phi_H(W') \leq \max_{0 \leq v \leq \mathsf{EL}^{\downarrow}(u)} F'(u, v, |X|, \mathsf{EL}, \mathsf{TL}) \tag{19}$$

$$\triangleq F'(u, |X|, \mathsf{EL}, \mathsf{TL}) \tag{20}$$

Proof: For any itemset W as described in Property 3, we have

$$\phi_H(W) = \frac{u|W|}{\sum_{t \in \mathcal{T}_W} |t|} \tag{21}$$

$$= \frac{u|W \cap (\mathcal{I} - Y)| + u|W \cap Y|}{\sum_{t \in \mathcal{T}_W} |t|} \tag{22}$$

$$= \frac{u|X| + uv}{\sum_{t \in \mathcal{T}_W} |t|} \tag{23}$$

Note that $\mathcal{T}_W$ are the supporting transactions of W and $|\mathcal{T}_W| = u$. Since $X \subset W$, by the anti-monotonicity of frequent itemsets, $\mathcal{T}_W \subseteq \mathcal{T}_X$. Since $|W \cap Y| = v$, for any $t \in \mathcal{T}_W$, $|t \cap Y| \geq |W \cap Y| = v$. Combining these two factors, $\sum_{t \in \mathcal{T}_W} |t| \geq \min_{l_1, \cdots, l_u, \mathsf{EL}(l_i) \geq v} \sum_{i=1}^{u} \mathsf{TL}^{\uparrow}(l_i)$. Thus,

$$\phi_H(W) = \frac{u|X| + uv}{\sum_{t \in \mathcal{T}_W} |t|} \leq \frac{u|X| + uv}{\min_{l_1, \cdots, l_u, \mathsf{EL}(l_i) \geq v} \sum_{i=1}^{u} \mathsf{TL}^{\uparrow}(l_i)}$$

For any itemsets W' in the subtree with $|\mathcal{T}_{W'}| = u$, it must be the case that $0 \leq |W' \cap Y| \leq \mathsf{EL}^{\downarrow}(u)$. We unify the bounds $F'(u, v, |X|, \mathsf{EL}, \mathsf{TL})$ over v and then have

$$\phi_H(W') \leq \max_{0 \leq v \leq \mathsf{EL}^{\downarrow}(u)} F'(u, v, |X|, \mathsf{EL}, \mathsf{TL})$$

■

Property 4: $F'(u+1, |X|, \mathsf{EL}, \mathsf{TL}) \leq F'(u, |X|, \mathsf{EL}, \mathsf{TL})$. Here, the definition of $F'(u, |X|, \mathsf{EL}, \mathsf{TL})$ is given in Equation (20). ■

Proof: By the definition of $F'(u, v, |X|, \mathsf{EL}, \mathsf{TL})$, similar to the proof of Inequality (16), we have

$$F'(u+1, v, |X|, \mathsf{EL}, \mathsf{TL}) \leq F'(u, v, |X|, \mathsf{EL}, \mathsf{TL})$$

for any u, v.

Since $\mathsf{EL}^{\downarrow}(u+1) \leq \mathsf{EL}^{\downarrow}(u)$, we have

$$F'(u+1, |X|, \mathsf{EL}, \mathsf{TL}) \tag{24}$$

$$= \max_{0 \leq v \leq \mathsf{EL}^{\downarrow}(u+1)} F'(u+1, v, |X|, \mathsf{EL}, \mathsf{TL}) \tag{25}$$

$$\leq \max_{0 \leq v \leq \mathsf{EL}^{\downarrow}(u+1)} F'(u, v, |X|, \mathsf{EL}, \mathsf{TL}) \tag{26}$$

$$\leq \max_{0 \leq v \leq \mathsf{EL}^{\downarrow}(u)} F'(u, v, |X|, \mathsf{EL}, \mathsf{TL}) \tag{27}$$

$$= F'(u, |X|, \mathsf{EL}, \mathsf{TL}) \tag{28}$$

■

With Properties 3 and 4 we can also easily prove that Theorems 1 and 2 also hold for $F'(u, |X|, \mathsf{EL}, \mathsf{TL})$.

Remark 1: The Functions of F and F' proposed in Property 1 and Property 3 can be both improved by a small technique. If $|t \cap Y| = |Y|$ for some $t \in \mathcal{T}_X$, then $Y \subseteq t$. Thus, t is a supporting transaction for any W in the subtree of X. This observation can further improve the two bounds above. We omit details because it would make the formulae too complicated. ■

Algorithm 3: The quality upper bound based on $F'(u, v, |X|, \mathsf{EL}, \mathsf{TL})$

input : the root set X, the extension set Y, the corresponding two vectors EL, TL and $n = |\mathcal{T}_X|$.
output: $qual$, the quality upper bound of any itemset in the subtree rooted at X.

```
1  sort ⟨EL, TL⟩ by ascending order of TL;
2  sum ← 0;
3  qual ← −∞;
4  for v ← 0 to EL↓(freq_min) do
5      u ← 0;
6      for i ← 1 to n do
7          if EL(i) ≥ v then
8              u ← u + 1;
9              sum ← sum + TL(i);
10             if u ≥ freq_min then
11                 occu ← (u|X| + u · v)/sum;
12                 qual ← max(qual, u/|T| + λ · occu);
```

Algorithm 3 shows the pseudo code for computing the quality upper bound based on $F'(u, v, |X|, \mathsf{EL}, \mathsf{TL})$. The time complexity

Quality Upper Bound	Complexity
F	$O(n + \mathsf{TL}^{\downarrow}(1))$
F'	$O(\mathsf{TL}^{\downarrow}(1) + n \cdot \mathsf{EL}^{\downarrow}(1))$
$\hat{F}$	$O(\mathsf{TL}^{\downarrow}(1) + mn)$

Table 4: The complexity of different estimation methods of quality upper bound

is $O(n + \mathsf{TL}^{\downarrow}(1) + n \cdot \mathsf{EL}^{\downarrow}(freq_{min}))$, in which $O(n + \mathsf{TL}^{\downarrow}(1))$ is for sorting the transactions (with counting sort algorithm) and $O(n \cdot \mathsf{EL}^{\downarrow}(freq_{min}))$ for the double loop in Line 4 and Line 6.

Also, we theoretically prove that F' is the tightest upper bound if only the values of $|X|$, EL and TL are used in the computation. Namely, we have the following theorem and its proof is omitted.

Theorem 3: $F'(u, |X|, \mathsf{EL}, \mathsf{TL})$ is the tightest upper bound for harmonic occupancy of any node in the subtree of X with u supporting transactions if we only use the values of $|X|$, EL, TL to compute the bound. ■

Corollary 1: $\max_{1 \le u \le |\mathcal{T}_X|} \left(\frac{u}{|\mathcal{T}|} + \lambda F'(u, |X|, \mathsf{EL}, \mathsf{TL}) \right)$ is the tightest upper bound of the harmonic quality (harmonic occupancy used inside) of any node in the subtree of X if we only use the values of $|X|, \mathsf{EL}, \mathsf{TL}$ to compute the bound. ■

4.2.3 *Tradeoff Between Bound Tightness and Computational Efficiency*

We have proposed two quality bounds based on harmonic occupancy so far. Their complexity is summarized in Table 4.2.3. Although F' is provably tighter than F, when $\mathsf{EL}^{\downarrow}(1)$ is large, it can also take 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 EL in the range of $[0, \mathsf{EL}^{\downarrow}(freq_{min})]$, we split this large interval into m smaller intervals $[v_0, v_1 - 1], \cdots, [v_{m-1}, v_m - 1]$ ($v_0 = 0, v_m = \mathsf{EL}^{\downarrow}(freq_{min}) + 1$).

For each interval $[v_{k-1}, v_k - 1]$, using the assumption that $v_{k-1} \le |W - X| < v_k$, we obtain a tighter bound on occupancy in Property 5. In the end, we unify the m bounds and get a final result. The time complexity for computing such a quality bound is $O(n + \mathsf{TL}^{\downarrow}(1) + mn)$. 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 4.

Property 5: For any itemset W in the subtree of X, let u be the frequency of W. Assume $v_k \le |W - X| < v_{k+1}$, then the harmonic occupancy of W satisfies that

$$\phi_H(W) \le \frac{u|X| + u \min(\mathsf{EL}^{\downarrow}(u), v_{k+1} - 1)}{\min_{l_1, \cdots, l_u, \mathsf{EL}(l_i) \ge v_k} \sum_{i=1}^{u} \mathsf{TL}^{\uparrow}(l_i)} \quad (29)$$

where t_{l_i} is any supporting transaction of X. ■

The proof is omitted here. Then, we have the following theorems.

Theorem 4: Let $u' = freq_{min}$ be the minimum frequency threshold. For any integers $0 = v_0 < v_1 < \cdots < v_m = \mathsf{EL}^{\downarrow}(u') + 1$,

$$\hat{F}(u', |X|, \mathsf{EL}, \mathsf{TL}) = \max_{0 \le k < m} \hat{F}(u', v_k, v_{k+1}, |X|, \mathsf{EL}, \mathsf{TL})$$

is the upper bound on harmonic occupancy for any frequent W in the subtree of X. And

$$\max_{u' \le u \le |\mathcal{T}_X|} \frac{u}{|\mathcal{T}|} + \lambda \hat{F}(u, |X|, \mathsf{EL}, \mathsf{TL})$$

is the upper bound on quality for any such W. ■

5. EVALUATION ON EFFECTIVENESS

In this section we evaluate the effectiveness of occupancy in a real-world application of print-area recommendation. Here, 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 (how to determine Q is omitted). 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 (for a given support threshold α and a occupancy threshold β).

To show the effectiveness of the proposed method we manually labeled the ground-truth of print-areas on the 2000 Web pages from the 100 major print-worthy Web sites (20 pages for each Web site).

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

For the 20 Web pages from a Web site, we use leave-one-out cross validation to evaluate the recommendation accuracy, i.e. we iteratively select one page as query and the log data on the left 19 Web pages are used to generate the transaction database for recommendation. A recommendation result actually refers to a set of content clips on the given Web page. Thus, we can evaluate its effectiveness by calculating the overlap area between the recommended clips and the ground truth on the query page. Then, with the overlap area we can calculate the precision P, recall R and $F1$ score of the recommendation in terms of area size. Specifically

$$P = \frac{|A_G \cap A_R|}{|A_R|}, R = \frac{|A_G \cap A_R|}{|A_G|}, F1 = 2 \times \frac{P \times R}{P + R}, \quad (30)$$

Where A_G is the clipping region of ground truth, A_R is the clipping region of the recommendation result and $|\dots|$ denotes the region size. If the precision is less than 1, it means that we need to remove some areas from the recommendation. If the recall is less than 1, it indicates that we need to add some contents to get the exact clipping areas. Then, we can average these these performance values over the 2000 Web pages to get the average performance.

The experiments in this section try to answer the following questions: (1) Does the concept of occupancy help to improve the recommendation performance compared with the baseline method? (2) How does the occupancy weight λ affect the recommendation performance?

The results of the baseline method are summarized in Table 5. The results of DOFIA are show in Table 6. Each entry in Tables 5 and 6 is the arithmetic average of the values (precision, recall and $F1$) over the 2000 Web pages. Since the $F1$ value is the harmonic average of precision and recall, the average $F1$ value over all those Web pages may be less than the average values of both precision and recall. For example, as shown in Table 6 when $\lambda = 0$ the average $F1$ value is 79.79%, which is smaller than the corresponding average values of precision and recall.

Note that for the baseline method, the maximal $F1$ score is 90.56% (when $\alpha = 0.1$), while DOFIA achieves a maximal $F1$ score of 93.8% (when $\lambda = 6.0$). So the improvement on recommendation accuracy is clear.

The role of λ in DOFIA 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 5.0, recommendation recall increases significantly 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 (see Equation (30)). However, when λ is too large, the performance of DOFIA 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.

6. EVALUATION ON EFFICIENCY

In this section we present the empirical evaluation on the efficiency of the proposed algorithm DOFIA over the large synthetic data sets. Specifically, we compare its running time with the baseline method to our problem. Here, the baseline method is to find all frequent itemsets first, compute the occupancy and quality for them, and then output top-k qualified ones. The implementation of MAFIA [6] includes a fast algorithm for frequent pattern mining, which is adopted in this comparison. We use MAFIA to find all the closed frequent itemsets, compute their quality values in the search process, and use a priority queue to maintain the top-k qualified itemsets. In practice, we find that the time to compute quality values and maintain the priority queue is actually negligible compared to the search process, so in the experiments below, we only show the time MAFIA used to find all the closed frequent itemsets, which is a lower bound for the time spent by the baseline. To abuse the notation a little bit, we still call the baseline method MAFIA. Our method, DOFIA, leverages the properties in Section 3 to further prune the search space, thus may achieve better efficiency.

The data sets used in this section are generated with the IBM synthetic data generator for itemset patterns [2]. We evaluate the efficiency on the data sets with different characteristics. The main parameters to generate the data include N: the number of transactions, L: the average length of transactions, I: the number of distinct items, PL: the average length of patterns, PN: the number of patterns. In our experiments, the default parameters for data generation are $N = 50000, I = 1000, L = 20, I = 1000, PL = 5, PN = 100$. In Section 6.1, We will adjust one parameter while fixing all the others to generate a series of data sets and show the efficiency changes. Only experiments on N and L are presented, and other parameters do not have much impact on running time of the algorithms when their values are within a reasonable range.

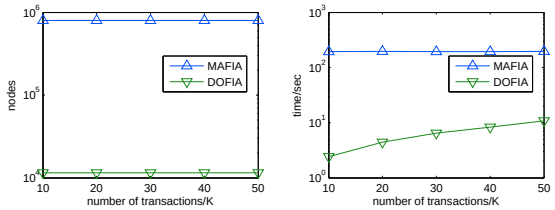
In addition, we also check the performance changes with different settings of problem parameters, including: the frequency threshold, α; the occupancy threshold, β; the number of top qualified itemsets we are searching for, k. The default values for these parameters are $\alpha = 0.005, \beta = 0.5, \lambda = 5, k = 5$. Similarly, we will adjust one parameter while fixing all the others to see the efficiency changes.

Table 5: The recommendation performance of the maximum frequent itemsets based method

α	*P(%)*	*R(%)*	*F1(%)*
0.0	85.85	96.02	88.74
0.05	90.28	92.2	89.81
0.1	90.6	92.84	**90.56**
0.2	90.65	92.12	90.05
0.3	89.5	91.02	88.88
0.4	87.11	87.57	85.48
0.5	82.0	81.75	79.06

Table 6: The recommendation performance ($\alpha = 0.05$ and $\beta = 0.1$)

λ	*P(%)*	*R(%)*	*F1(%)*
0.0	90.04	82.15	79.79
0.5	89.67	92.84	88.78
1.0	90.77	94.74	91.3
2.0	91.63	95.96	92.81
4.0	92.65	96.31	93.6
5.0	92.81	96.3	93.64
6.0	93.23	96.19	**93.8**
8.0	93.23	95.95	93.7
10.0	93.34	95.84	93.71
$+\infty$	91.27	91.62	89.82
Average	91.76	93.91	91.12



(a) Number of Nodes (b) Running Time

Figure 2: The effects of the number of transactions on MAFIA and DOFIA

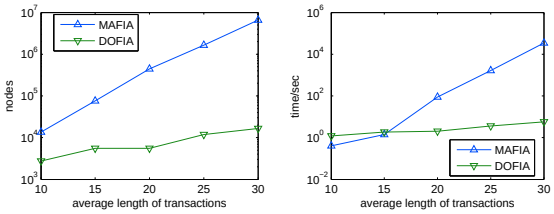
Finally, we will compare the tradeoff between computational efficiency and bound tightness. By default, we use the upper bounds proposed in Section 4.2.1.

The experiments were performed on a Windows 7 laptop with quad core Intel i5-540M processor and 4 GB of main memory.

6.1 Evaluation on Different Databases

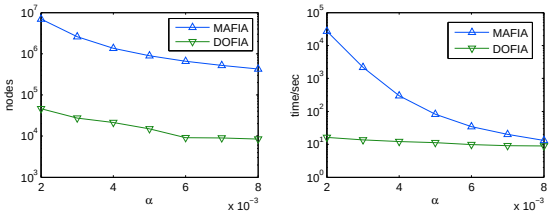
The number of transactions. Here, we generate a database of $N = 10000$ transactions and scale it up by vertical concatenation of the database. The results are shown in Figure 2, including the running time and the number of nodes in the subset tree searched by MAFIA and DOFIA. Note that in this experiments we only duplicate the database to increase its size. Thus, the number of nodes visited in the subset tree are expected to stay unchanged. As can be seen from Figure 2(a), DOFIA only searches about 1.4% nodes of those searched by MAFIA. With respect to running time, DOFIA's running time grows linearly, from 2.45 seconds for 10000 transactions to 10.34 seconds for 50000 transactions. What is interesting is that MAFIA's running time remains stable when the number of transactions grows. After careful investigation, we think it is due to the extreme efficiency of bit operators used extensively by MAFIA. However, since it takes MAFIA about 195 seconds to run the experiment for each data set, DOFIA is still much faster.

The average length of transactions. Here, we vary the average length of transactions. With the number of items growing in each transaction, the length of interesting patterns also grows. So we expect an exponentially growing number of frequent itemsets. Without proper pruning, MAFIA obviously cannot handle such cases. As shown in Figure 3, although it is faster than DOFIA when $L = 10$, it becomes very slow as L grows larger, taking more than 10^6 seconds when $L \geq 25$. On the hand, with the help of efficient pruning on occupancy and quality, DOFIA's running time grows smoothly, from 1.22 seconds for $L = 10$ to 5.88 seconds for $L = 30$.



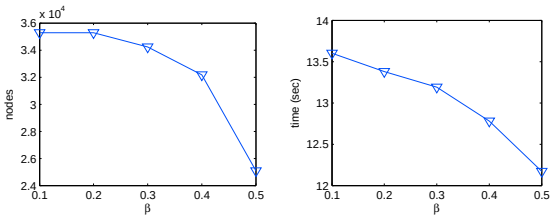
(a) Number of Nodes (b) Running Time

Figure 3: The effects of the average length of transactions on MAFIA and DOFIA



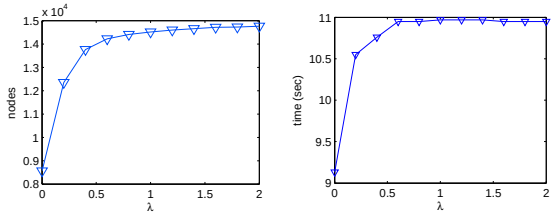
(a) Number of Nodes (b) Running Time

Figure 4: The effects of the minimum frequency threshold on MAFIA and DOFIA



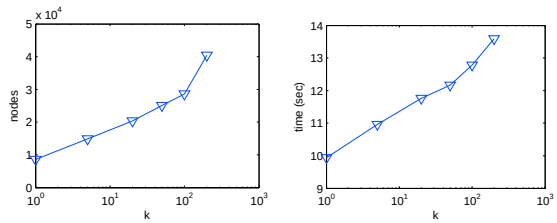
(a) Number of Nodes (b) Running Time

Figure 5: The effects of the minimum occupancy threshold on DOFIA



(a) Number of Nodes (b) Running Time

Figure 6: The effects of the occupancy weight parameter on DOFIA



(a) Number of Nodes (b) Running Time

Figure 7: The effects of top-k on DOFIA

6.2 Evaluation on Different Problem Settings

The frequency threshold. In practice, it is often desirable to have a low frequency threshold in order to detect more diverse patterns. However, with the lowering of frequency also comes the problem of too many frequent patterns and long running time. As seen in Figure 4, although MAFIA is just a little slower than DOFIA when frequency threshold $\alpha = 0.8\%$, it running time grows quickly with the decrease on α, and takes longer than 10000 seconds when $\alpha = 0.2\%$. On the other hand, DOFIA remains relatively efficient even at very low frequency threshold, due to the pruning on occupancy and quality.

The occupancy threshold. As shown in Figure 5, with the decrease of the occupancy threshold β, the running time and the number of the visited nodes increases moderately for DOFIA. As beta β decreases from 0.5 to 0.1, the number of nodes searched increases by 40.6% and the running time increases by 11.1%. One might expect that the increase in running time is exponential, but since we are doing the top-k qualified search, the most qualified nodes DOFIA 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.

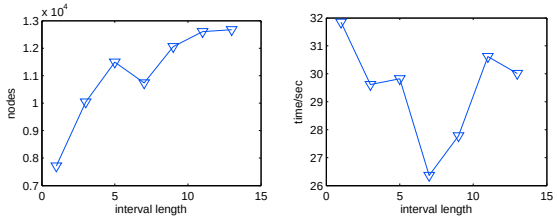
The occupancy weight parameter. The occupancy weight parameter λ reflects the priority we put on occupancy. Here, we empirically test DOFIA's performance when we vary λ. As can be seen from Figure 6, with the increase of λ from 0 to 2, the number of nodes visited by DOFIA increased by 80%, and the running time increased by 17%. The increase is mainly due to the fact that when λ is small the quality is dominated by its support, thus the high-frequency itemset with a small number of items will be output very fast. When λ becomes larger, it has to visit much more itemsets with high occupancy. When $\lambda > 1$, 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. With more patterns we can even do ranking on these patterns and find a set of diverse and high-quality itemsets. Here, we increase k to check the performance of DOFIA. As can be seen from Figure 7, though the number nodes grows about 300% when k increases from 1 to 200, the running time only increases moderately by about 40%.

6.3 Tradeoff between Bound Tightness and Computational 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 $|W \cap Y|$ in the range of $[0, \mathsf{EL}^{\downarrow}(freq_{min})]$ as F' does, we suggest to split the large interval into smaller intervals $[v_0, v_1 - 1], \cdots, [v_{K-1}, v_K - 1]$, $(v_0 = 0, v_K = \mathsf{EL}^{\downarrow}(freq_{min}) + 1)$.

Figure 8 shows the effect of this technique in one of our synthetic data sets. The x-axis shows the "interval length", i.e. $v_i - v_{i-1}$ and the y-axis shows the number of nodes searched and the running time in the two figures respectively. As can be seen, smaller the interval length, tighter the bound and fewer the nodes searched. In the case where the interval length is 1, only 60% nodes are searched compared to the case where the interval length is 13. However, the shortest running time does not occur when the bound is tightest, since the computation is too costly. In this case, the algorithm is fastest when the interval length is 7.



(a) Number of Nodes (b) Running Time

Figure 8: Tradeoff between computing efficiency and bound tightness

7. RELATED WORKS

Frequent pattern mining [1] has been well recognized to be fundamental to many important data mining tasks. There is a great amount of work that studies efficient mining of frequent patterns [11, 9, 6, 12], and we refer to [10] as a recent review on this field. These algorithms can be classified into mining frequent patterns [11, 9], frequent maximal patterns [6], and frequent closed patterns [12]. To reduce the number of frequent patterns some interestingness measures and constraints are proposed [13, 4, 5, 8] along with algorithms to implement them efficiently. Recently, graphical models[16] and compression[15] have also been proposed to approximate the frequent itemset with a minimal number of rules or patterns.

The concept of *occupancy* proposed in this paper can be viewed as a new interesting measure and constraint. It can be seen as the *relative size* of a pattern to its supporting transactions rather than the *absolute size* of it, and might be more meaningful in many applications. After proposing the concept of occupancy, we then formulate the problem of mining top-k qualified patterns which maximize the weighted sum of support and occupancy. Empirical tests on web print recommendation shows that the top qualified patterns can significantly improve the recommendation results when the weight on occupancy is chosen properly.

One work very similar to the spirits of ours is [17], where Wang et al. formulated the problem of finding the top-k frequent itemsets with their sizes no smaller than a threshold. Here we should note the difference between our work and theirs. First, [17] used the absolute size of an itemset as a constraint, while we use occupancy, the relative size of an itemset to its supporting transactions. As argued before, in many cases occupancy might be a more reasonable constraint. Besides, since occupancy is a normalized value, it also makes the tuning easier. Second, [17] tried to find the top-k frequent itemset, 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. In another work [8], Gade et al. proposed to maximize the product of itemset size and its frequency. Again, occupancy might be a better measure than absolute itemset size. Recent works [7, 14] 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 in might be an interesting work to extend these frameworks to handle occupancy properly.

Different from many other constraints proposed, occupancy is not *anti-monotonic*, *monotonic*, *convertible*, and *succinct* [8], thus, no previous methods can be leveraged.

8. CONCLUSION

In this study, motivated by the pattern recommendation applications we introduce a new measure of pattern interestingness *occupancy* and formulate the problem of mining top-k qualified patterns. To solve this problem, we explore the upper bound properties on occupancy and propose DOFIA that injects these properties deeply into the search process. We propose two upper bounds, in which the first one is more efficient and the second one is theoretically proved to be tightest with certain input constraints. We show the effectiveness of occupancy in the real-world application of print-area recommendation for Web pages. Finally, on large synthetic data we demonstrate that DOFIA significantly outperforms the baseline method in terms of efficiency. It is worth mentioning that the concept of occupancy can be extended to sequential pattern mining [3] and graph mining [18], and thus is useful in many pattern mining applications.

9. REFERENCES

[1] R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In *Proceedings of the ACM SIGMOD*, 1993.

[2] R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, and A. I. Verkamo. Advances in knowledge discovery and data mining. chapter Fast discovery of association rules, pages 307–328. American Association for Artificial Intelligence, Menlo Park, CA, USA, 1996.

[3] R. Agrawal and R. Srikant. Mining sequential patterns. In *Proceedings of the 11th ICDE*, 1995.

[4] F. Bonchi, F. Giannotti, A. Mazzanti, and D. Pedreschi. Exante: Anticipated data reduction in constrained pattern mining. *Knowledge Discovery in Databases: PKDD 2003*, 2003.

[5] F. Bonchi and C. Lucchese. On closed constrained frequent pattern mining. In *Data Mining, 2004. ICDM'04. Fourth IEEE International Conference on*, pages 35–42. IEEE, 2004.

[6] D. Burdick, M. Calimlim, and J. Gehrke. Mafia: A maximal frequent itemset algorithm for transactional databases. In *Proceedings of the 17th ICDE*, 2001.

[7] L. Cerf, J. Besson, C. Robardet, and J.-F. Boulicaut. Closed patterns meet n-ary relations. *ACM Trans. Knowl. Discov. Data*, 3(1):3:1–3:36, Mar. 2009.

[8] K. Gade, J. Wang, and G. Karypis. 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*, pages 138–147. ACM, 2004.

[9] K. Gouda and M. Zaki. Genmax: An efficient algorithm for mining maximal frequent itemsets. *Data Mining and Knowledge Discovery*, 2005.

[10] J. Han, H. Cheng, D. Xin, and X. Yan. Frequent pattern mining: current status and future directions. *Data Mining and Knowledge Discovery*, 2007.

[11] J. Han, P. Jian, and Y. Yin. Mining frequent patterns without candidate generation. In *Proceedings of ACM SIGMOD*, 2000.

[12] N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal. Discovering frequent closed itemsets for association rules. In *Proceedings of the 7th International Conference on Database Theory*, 1999.

[13] J. Pei, J. Han, and L. Lakshmanan. Mining frequent itemsets with convertible constraints. In *Proceedings of the 17th ICDE*, 2001.

[14] A. Soulet and B. Crémilleux. Mining constraint-based patterns using automatic relaxation. *Intell. Data Anal.*, 13(1):109–133, Jan. 2009.

[15] J. Vreeken, M. van Leeuwen, and A. Siebes. Krimp: mining itemsets that compress. *Data Mining and Knowledge Discovery*, pages 1–46, 2011.

[16] C. Wang and S. Parthasarathy. Learning approximate mrfs from large transactional data. *Statistical Network Analysis: Models, Issues, and New Directions*, pages 182–185, 2007.

[17] J. Wang, J. Han, Y. Lu, and P. Tzvetkov. Tfp: An efficient algorithm for mining top-k frequent closed itemsets. *Knowledge and Data Engineering, IEEE Transactions on*, 2005.

[18] T. Washio and H. Motoda. State of the art of graph-based data mining. *ACM SIGKDD Explorations Newsletter*, 2003.