DOI 10.1007/s10115-003-0133-6

Knowledge and Information Systems (2004)

Knowledge and Information Systems

Mining Condensed Frequent-Pattern Bases*

Jian Pei[1], Guozhu Dong[2], Wei Zou[3], Jiawei Han[3]

[1]State University of New York at Buffalo, USA
[2]Wright State University, USA
[3]University of Illinois at Urbana-Champaign, USA

Abstract. Frequent-pattern mining has been studied extensively and has many useful applications. However, frequent-pattern mining often generates too many patterns to be truly efficient or effective. In many applications, it is sufficient to generate and examine frequent patterns with a sufficiently good approximation of the support frequency instead of in full precision. Such a compact but "close-enough" frequent-pattern base is called a *condensed frequent-pattern base*.

In this paper, we propose and examine several alternatives for the design, representation, and implementation of such condensed frequent-pattern bases. Several algorithms for computing such pattern bases are proposed. Their effectiveness at pattern compression and methods for efficiently computing them are investigated. A systematic performance study is conducted on different kinds of databases, and demonstrates the effectiveness and efficiency of our approach in handling frequent-pattern mining in large databases.

Keywords: Frequent patterns; Frequent-pattern mining; Approximation; Compression

1. Introduction

It has been well recognised that frequent-pattern mining plays an essential role in many important data mining tasks, such as mining association (Agrawal and Srikant 1994), correlation (Brin et al. 1997), causality (Silverstein et al. 1998), sequential patterns (Agrawal and Srikant 1995; Pei et al. 2001), episodes (Mannila et al. 1997), multi-dimensional patterns (Lent et al. 1997; Kamber et al. 1997), max-patterns (Bayardo 1998), partial periodicity (Han et al. 1999), and emerging patterns (Dong and Li 1999). Frequent-pattern mining techniques have also been

* This paper is a substantially enhanced and enriched version of the paper "*On Computing Condensed Frequent Pattern Bases*", by Jian Pei, Guozhu Dong, Wei Zou, and Jiawei Han, appearing in the Proceedings of the IEEE 2002 International Conference on Data Mining, December, 2002, Japan.

Received 15 September 2002
Revised 20 December 2002
Accepted 24 January 2003
Published online 9 February 2004

shown to be useful in some other tasks, such as iceberg-cube computation (Beyer and Ramakrishnan 1999) and classification (Liu et al. 1998). However, it has also been widely recognised that frequent-pattern mining often produces a huge number of patterns (Zaki 2000), which reduces not only the efficiency but also the effectiveness of mining, since it is unrealistic to store and comprehend so many patterns.

Recently, efforts have been devoted to address this problem. In general, interesting proposals can be classified into two categories. Firstly, concise representations of frequent patterns have been explored, such as *frequent closed patterns* (Pasquier et al. 1999; Zaki 2000; Pei et al. 2000), which can be used to remove sub-patterns which have the same support as some of their super-patterns. Studies such as (Zaki 2000) have shown that by doing so, the total number of patterns and rules can be reduced substantially, especially in dense data sets. Secondly, constraints can be used to capture the users' focus, and effective strategies have been developed to push various constraints deep into the mining process (Ng et al. 1998; Lakshmanan et al. 1999; Pei et al. 2001).

Even though these approaches are useful, they may still be insufficient in some situations. Compression using the closed-pattern approach may not be very effective, since slightly different counts often exist between super- and sub-patterns. Constraint-based mining, though useful, can hardly be used for pre-computation, since different users are likely to have different constraints.

Although it seems to be inherent that a large database will contain numerous frequent patterns, it is easy to observe a simple fact in practice: *Most applications will not need* **precise** *support information for frequent patterns: a* **good approximation** *for the support count could be more than adequate.* Here, by a *good approximation*, we mean that the frequency of every frequent pattern can be estimated with a *guaranteed maximal error bound*. For example, for a frequent pattern {diapers, beer}, instead of giving the exact support count (e.g., 10 000), a range, e.g., $10\,000 \pm 1\%$, may be good enough. Here 1% indicates the *error bound* for the range.

Using the approximations of frequent patterns, one can derive approximate association rules. By introducing approximations, many redundant rules can be removed and the number of rules can be reduced substantially. For example, suppose that if a customer buys {bread, milk}, the probability that s/he also buys {cheese, cereal, butter} is almost the same as the probability that s/he buys cereal. We may have two rules, R_1 : {*bread, milk*} $\rightarrow$ {*cheese, cereal, butter*} and R_2 : {*bread, milk*} $\rightarrow$ {*cereal*}, both with approximate support $2.5\% \pm 0.1\%$ and approximate confidence $85\% \pm 1\%$. In this case it is obvious that rule R_2 can be pruned. With a reasonable error bound, it is likely that many similar sub-rules can be pruned.

As many studies have shown, frequent patterns can be used effectively in many other data mining tasks, such as association-based classification (Liu et al. 1998; Li et al. 2001), frequent-pattern-based clustering (Beil et al. 2002), and multidimensional cube or transaction gradient analysis (Imielinski et al. 2002; Dong et al. 2001). Approximate frequent-pattern mining will benefit these tasks as well. For example, approximate frequent patterns may substantially reduce the number of rules to be stored in a classifier, which not only leads to efficient construction of a compact association-rule-based classifier, but also speeds up the prediction process.

We believe that a condensed frequent-pattern base is not only acceptable but often more preferable in applications for the following reasons.

- *When mining a large database, a small deviation often has a minor effect on the analysis*. For an analyst, the exact information "*diapers and beer have been*

bought together 10 050 *times out of the* 10 *million transactions*" and an approximation "*diapers and beer have been bought together* $10\,050 \pm 50$ *times*" may not have any essential difference. Notice that such a minor deviation will usually have to be ignored when truncation or rounding is needed for presentation to users. Thus, what an analyst really cares about is that the mined patterns are close enough and a specified error bound is guaranteed.

- *Condensing the frequent-pattern base leads to more effective frequent-pattern mining*. By computing a condensed pattern base, the number of patterns can be reduced dramatically, but the general information of the frequent patterns is essentially preserved. A much smaller base of patterns certainly helps users comprehend the mining results.
- *Computing a condensed frequent-pattern base leads to more efficient frequent-pattern mining*. A condensed frequent pattern base could be much smaller than the complete frequent pattern base. Thus, one may need to compute and access a much smaller pattern base, which leads to better efficiency.

In summary, mining a condensed frequent-pattern base may make frequent-pattern mining more realistic in real-life applications.

In this paper, the concept of *condensed frequent-pattern bases with guaranteed maximal error bound* is introduced and methods for the *efficient computation of such a condensed pattern base* are studied. More specifically, this paper makes the following contributions.

1. *We introduce the concept of the* **condensed frequent pattern base** *and devise systematic representations of such frequent-pattern bases*. We show that such representations achieve satisfactory approximations with a guaranteed maximal error bound on the support.
2. *We develop efficient algorithms for computing condensed pattern bases directly from transaction databases.* Our algorithms are able to prune many patterns in the mining process, by exploiting the relaxation of the counting requirement allowed by frequent-pattern bases.
3. *We present a systematic performance study to verify the effectiveness and efficiency of condensed frequent-pattern bases*. Our results show that computing condensed frequent-pattern bases is highly promising as a practical frequent-pattern mining approach for large databases.

Previously, the ideas of approximating frequent patterns have been probed in some related studies. For example, (Mannila and Toivonen 1996) showed that approximate association rules are interesting and useful. In (Boulicaut et al. 2000), the notion of free-sets was proposed and led to an error-bound approximation of frequencies. However, none of the previous studies systematically explored the problem of designing and mining condensed frequent-pattern bases with a guaranteed maximal error bound.

The rest of this paper is organised as follows. The problem of computing a condensed frequent-pattern base is introduced in Sect. 2. A level-by-level frequent-pattern base construction method is presented in Sect. 3. In Sect. 4, we develop an effective and efficient method for frequent-pattern base construction using max-patterns at various layers. Section 5 presents a comprehensive performance study to demonstrate the effectiveness and efficiency of our approach. We discuss related issues, potential extensions, implications, and applications in Sect. 6, and conclude our study in Sect. 7.

2. Problem Definition

We firstly review some standard terminology for frequent-pattern mining. Let $I = \{i_1, \ldots, i_n\}$ be a set of literals, called *items*. An *itemset* (or *pattern*) X, denoted by $X = i_{j_1} \cdots i_{j_l}$ (i.e., by omitting set brackets), is a subset of items in I. An itemset with l items is called an *l-itemset*. For two patterns X and Y such that $X \subseteq Y$, Y is called a *super-pattern* of X, and X a *sub-pattern* of Y.

A *transaction* $T = (tid, X)$ is a tuple in which *tid* is a *transaction identifier* and X is an itemset. A transaction $T = (tid, X)$ is said to *contain* itemset Y if $Y \subseteq X$. A *transaction database* TDB is a set of transactions. The *support* of an itemset X in TDB, denoted by $sup(X)$, is the number of transactions in TDB containing X, i.e., $sup(X) = |\{(tid, Y)|((tid, Y) \in TDB) \wedge (X \subseteq Y)\}|$.

Given a transaction database TDB and a *support threshold* min_sup, an itemset X is called a *frequent itemset* or a *frequent pattern* if $sup(X) \geq min\_sup$. The problem of frequent-pattern mining is to find the complete set of frequent patterns from TDB w.r.t. a user-specified support threshold min_sup. The set of all frequent patterns is called a *frequent-pattern base*, or *FP-base* for short.

It is often expensive to find the complete set of frequent patterns, since an FP-base may contain a huge number of frequent patterns. For example, in a transaction database TDB containing only one transaction, $(1, a_1 \cdots a_{100})$, every non-empty sub-pattern of $a_1 \cdots a_{100}$ is a frequent pattern. Thus, the FP-base has $(2^{100} - 1) \approx 10^{30}$ frequent patterns!

In this paper, we propose to overcome the difficulty caused by the huge number of frequent patterns as follows: compute a small subset of frequent patterns, i.e., a *condensed FP-base*, and then use it to approximate the supports of arbitrary frequent patterns.

Problem statement. Given a transaction database, a support threshold, and a user-specified *error bound* k, the *problem of computing a condensed FP-base* is to find a subset of frequent patterns $\mathcal{B}$ and a function $f_{\mathcal{B}}$ such that the following holds for each pattern X:

$$f_{\mathcal{B}}(X) = \begin{cases} 0 & \text{if } X \text{ is infrequent}, \\ [sup_{lb}, sup_{ub}] \text{ s.t. } (sup_{lb} \leq sup(X) \leq sup_{ub}) \text{ and } (sup_{ub} - sup_{lb}) \leq k & \text{if } X \text{ is frequent}. \end{cases}$$

The function $f_{\mathcal{B}}$ is called a *(support) approximation function*, and the set $\mathcal{B}$ is called a *condensed FP-base* w.r.t. $f_{\mathcal{B}}$.[1]

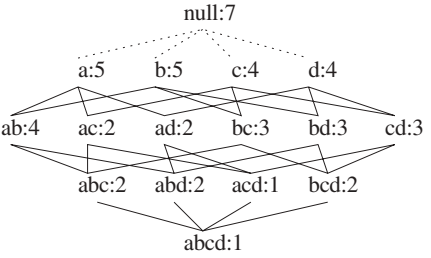
Example 1. Consider the transaction database shown in Table 1. Let the support threshold be $min\_sup = 1$ and the error bound be $k = 2$. The lattice of a total of 15 frequent patterns is shown in Fig. 1a.

The set $\mathcal{B}_d = \{a : 5, b : 5, c : 4, d : 4, acd : 1, abcd : 1\}$ is a *condensed FP-base*. Patterns in $\mathcal{B}_d$ are those labelled with supports in Fig. 1b. For each pattern X, the

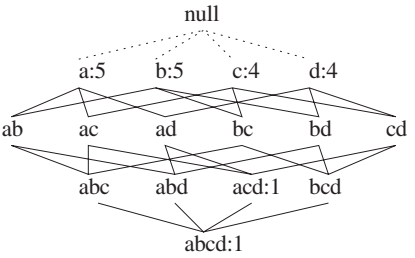
[1] Instead of an absolute error bound k, computing a condensed FP-base can also take a relative, percentage-based error bound $k\%$. In that case, $\frac{sup_{ub} - sup_{lb}}{sup_{lb}} \leq k\%$ should be satisfied for frequent patterns.

Table 1. A transaction database with seven transactions.

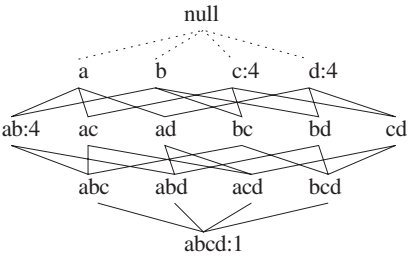
Transaction-id	Itemset
10	*a*
20	*ab*
30	*abc*
40	*abcd*
50	*cd*
60	*abd*
70	*bcd*



(a) Lattice of frequent patterns



(b) Condensed FP-base $\mathcal{B}_d$



(c) Condensed FP-base $\mathcal{B}_m$

Fig. 1. Lattice of frequent patterns for Example 1.

function $f_{\mathcal{B}_d}$ is defined as follows:

$$f_{\mathcal{B}_d}(X) = \begin{cases} 0 & \text{if there exists no } X' \in \mathcal{B}_d\,, \\ & \quad \text{s.t. } X \subseteq X' \\ [sup(X), sup(X)] & \text{if } X \in \mathcal{B}_d\,, \\ [sup(X_0) - 2, sup(X_0)] & X_0 \subset X \text{ and} \\ & \quad sup(X_0) = \min(sup(X'')) \\ & \quad \text{for } X'' \subset X \text{ and } X'' \in \mathcal{B}_d\,. \end{cases}$$

For example, $f_{\mathcal{B}_d}(abcde) = 0$ for the infrequent pattern *abcde*, since there is no $X' \in \mathcal{B}_d$ s.t. $abcde \subseteq X'$; $f_{\mathcal{B}_d}(ac) = [4-2, 4] = [2, 4]$, since c is a sub-

pattern of ac in $\mathcal{B}_d$ with the smallest support count (of 4). Here, we used the well-known a priori property that $supp(X) \geq supp(Y)$ if $X \subseteq Y$. One can verify that $f_{\mathcal{B}_d}$ can approximate the support count of each frequent pattern as required by the definition given above. For example, $sup(ab)$ is approximated by $[3, 5]$ and $sup(abc)$ by $[2, 4]$.

Moreover, $\mathcal{B}_m = \{c : 4, d : 4, ab : 4, abcd : 1\}$ is another condensed FP-base, as plotted in Fig. 1c. The corresponding approximation function $f_{\mathcal{B}_m}$ is defined (for each pattern X) as follows:

$$f_{\mathcal{B}_m}(X) = \begin{cases} 0 & \text{if there exists no } X' \in \mathcal{B}_m, \\ & \quad \text{s.t. } X \subseteq X' \\ [sup(X), sup(X)] & \text{if } X \in \mathcal{B}_m, \\ [sup(X_0), sup(X_0) + 2] & X_0 \supset X \text{ and } sup(X_0) = \\ & \quad \max\{sup(X'') \mid X'' \supset X, X'' \in \mathcal{B}_m\}. \end{cases}$$

Condensed FP-bases and approximation functions are not unique. A superset of a condensed FP-base is also a base w.r.t. the identical approximation function. A condensed FP-base is *minimal* (w.r.t. an approximation function f) if it does not contain a proper subset which is also a condensed FP-base w.r.t. f. Interestingly, even minimal condensed FP-bases are not unique. For Example 1, both $\mathcal{B}_d$ and $\mathcal{B}_m$ are *minimal* condensed FP-bases.

Among possible approximation bases, we prefer those requiring as little space as possible. Such condensed FP-bases offer a significant compression effect, which can be measured by the *compression ratio*, defined as

$$\text{compression ratio} = \frac{\text{number of patterns in the condensed FP-base}}{\text{total number of frequent patterns}}. \quad (1)$$

Clearly, the smaller the compression ratio, the better the compression effect. We observe from Example 1 that condensed FP-bases can produce considerable space savings even with a small error bound. ($\mathcal{B}_d$ achieves a compression ratio of 40%, whereas $\mathcal{B}_m$ achieves 26.7%. $\mathcal{B}_m$ achieves better compression than $\mathcal{B}_d$.)

Previous research has also considered computing reduced sets of frequent patterns, including reduction based on frequent closed itemsets (Pasquier et al. 1999) and containment-based reduction (Bayardo 1998; Dong and Li 1999). An itemset X is called a *closed pattern* if there exists no proper superset X' of X such that $sup(X) = sup(X')$, while X is called a *max-pattern* if there exists no superset X' of X such that X' is also frequent. Interestingly, it can be shown that the complete set of frequent closed patterns is a minimal condensed FP-base with error bound 0, while the complete set of max-patterns is a minimal condensed FP-base with error bound $(|TDB| - min\_sup)$, where min_sup is the support threshold. However, none of these considers approximating supports of frequent patterns with a user-specified error bound as we do here.

Now, let us ask the following question: *How can we construct condensed FP-bases effectively and efficiently?* This is the topic of the following sections. We will consider two approaches: the first method (see Sect. 3) constructs a base by considering all frequent patterns in the a priori manner; the second method (see Sect. 4) constructs a base by considering only maximal frequent patterns at various layers for a range of support thresholds.

3. Constructing a Condensed FP-Base Level-by-Level

We now consider an approach that constructs a condensed FP-base by examining all frequent patterns level-by-level: *A frequent pattern is added into the condensed FP-base only if it cannot be approximated by its sub-patterns in the base*. The method is illustrated in the following example.

Example 2. We construct a condensed FP-base $\mathcal{B}_d$ as follows for the transaction database *TDB* in Table 1, for a support threshold of 1 and an error bound of 2. $\mathcal{B}_d$ is shown in Fig. 1b, while the approximation function $f_{\mathcal{B}_d}$ is defined in Example 1.

For each pattern X, let $X.ub$ denote $\min\{sup(X') | X' \in \mathcal{B}_d$, and $X' \subseteq X\}$, i.e., $X.ub$ is the minimum of the supports of all proper sub-patterns of X in $\mathcal{B}_d$.

1. We initialise $\mathcal{B}_d = \emptyset$ and mine length-1 and length-2 frequent patterns. Since length-1 frequent patterns are the *most frequent end borders* of frequent patterns and none of their sub-patterns is in the base, we insert all of them (i.e., a, b, c, and d) into $\mathcal{B}_d$.
   For each length-1 frequent pattern x, $x.ub = sup(x)$.
2. For the next level, i.e., length-2 frequent patterns, we have $xy.ub = \min(x.ub, y.ub)$ for each length-2 frequent pattern xy.
   We do two types of insertion into $\mathcal{B}_d$. Firstly, a length-2 frequent pattern X is added into $\mathcal{B}_d$ if $X.ub - sup(X)$ is over the error bound (i.e., if its support cannot be approximated by its sub-patterns in the base $\mathcal{B}_d$). In this example, since all length-2 frequent patterns can be approximated properly by their length-1 sub-patterns, no length-2 frequent pattern is inserted into $\mathcal{B}_d$.
   Secondly, if a length-2 frequent pattern has no frequent length-3 super-pattern, i.e., it is a max-pattern, then it is inserted into $\mathcal{B}_d$. Max-patterns are needed in $\mathcal{B}_d$ since they are used to determine whether a pattern is frequent. In this example, no such length-2 frequent pattern exists.
3. For the length-3 level, since $acd.ub - sup(acd) = 4 - 1 > 2$, pattern acd is inserted into $\mathcal{B}_d$. Here, $abc.ub = \min(ab.ub, ac.ub, bc.ub)$. After the insertion, we set $abc.ub = sup(abc) = 1$. We then mine length-4 frequent patterns and see that there is no length-3 max-pattern.
4. The length-4 frequent pattern $abcd$ is a max-pattern, since there are no length-5 frequent patterns, and so it is inserted into $\mathcal{B}_d$.

At the end, the base $\mathcal{B}_d$ contains 6 patterns: $\{a : 5, b : 5, c : 4, d : 4, acd : 1, abcd : 1\}$. Since the search is downward from length-1 patterns, we call the resulting base a *downward condensed FP-base*.

Now, let us generalise the level-by-level condensed FP-base construction method. We firstly define the approximation function ϑ as follows.

Definition 1. Given a condensed FP-base $\mathcal{B}$, an error bound k and a pattern X:

$$\vartheta(X) = \begin{cases} 0 & \text{if there exists no } X' \in \mathcal{B} \text{ s.t. } X' \supseteq X\,, \\ [sup(X), sup(X)] & \text{if } X \in \mathcal{B}\,, \\ [m-k, m] & \text{if } X \notin \mathcal{B}, \text{ where} \\ & \quad m = \min\{sup(X') \mid X' \in \mathcal{B}, X' \subset X\}\,. \end{cases}$$

The following algorithm computes a condensed FP-base with respect to approximation function ϑ.

Algorithm 1 (*CFP-D*: a level-by-level downward search method).

Input: A transaction database TDB, the support threshold min_sup, and the error bound k
Output: A condensed FP-base $\mathcal{B}$ w.r.t. ϑ
Method:

1: let $\mathcal{B} = \emptyset$;
2: find length-1 frequent patterns and insert them into $\mathcal{B}$;
3: for each length-1 frequent pattern X, let $X.ub = sup(X)$;
4: let $i = 1$;
5: repeat
6: i = i+1;
7: generate the set $\mathcal{F}_i$ of length-i frequent patterns;
8: for each length-i frequent pattern X,
let $X.ub = \min(X'.ub)$, where X' ranges over length-$(i-1)$ sub-patterns of X;
// the calculation of $X.ub$ can be done as a byproduct of
// candidate-generation
9: if $(X.ub - sup(X)) > k$, then insert X into $\mathcal{B}$ and set $X.ub = sup(X)$;
10: for each length-$(i-1)$ frequent pattern X s.t. X has no super-pattern in $\mathcal{F}_i$, insert X into $\mathcal{B}$;
// rationale: X is a max-pattern
11: until $\mathcal{F}_i = \emptyset$;
12: return $\mathcal{B}$

One advantage of the method shown in Example 2 is that it is intuitive and can be easily integrated into Algorithm 1. The correctness and effectiveness of the algorithm are shown in the following theorem.

Theorem 3.1. Algorithm 1 returns a minimal condensed FP-base with respect to approximation function ϑ.

Proof. The fact that the algorithm returns a condensed FP-base w.r.t. ϑ can be proven by induction on the pattern length i. From steps 1–3, we see that all length $i = 1$ frequent patterns can be properly approximated, since they belong to $\mathcal{B}$. For length i pattern X for $i > 1$, two cases arise. Case (a): X is not a maximal frequent pattern (i.e., it is contained in some other frequent patterns). Then, let $X.ub$ be the minimum of the supports of X's sub-patterns in $\mathcal{B}$. We need to insert X into $\mathcal{B}$ if the difference between $X.ub$ and X's support is more than k, since X's support cannot be approximated properly. This is exactly what step 9 does. Case (b): X is a maximal frequent pattern. Then we need to insert X into $\mathcal{B}$; this is exactly what step 10 does.

It is clear that if we remove any pattern X from $\mathcal{B}$, ϑ cannot give an approximation for $sup(X)$ under the error bound. Thus, $\mathcal{B}$ is minimal.

What kind of patterns are included in $\mathcal{B}$ computed by Algorithm 1? A frequent pattern X is called a *seed pattern* if for each proper sub-pattern $X' \subset X$, $sup(X') > sup(X)$. Interestingly, we have

Lemma 3.1. Every pattern in $\mathcal{B}$ computed by Algorithm 1 is either a seed pattern or a max-pattern.

Proof. Let $X \in \mathcal{B}$ be a pattern which is not a seed pattern. Then there exists another pattern $X' \subset X$ such that $sup(X) = sup(X')$. Clearly, $X'.ub \geq X.ub$. Thus, $X'.ub - sup(X') \geq X.ub - sup(X)$. Since $X \in \mathcal{B}$, it follows that $X' \in \mathcal{B}$. It is easy to see that $X.ub = sup(X')$, and hence $X.ub - sup(X) = 0$. Since X is inserted into $\mathcal{B}$, X is a max-pattern.

4. Constructing a Condensed FP-Base Using Max-Patterns

While Algorithm 1 is intuitive and correct, it has to check every frequent pattern. When there are many frequent patterns, the mining cost is non-trivial. *Can we avoid checking every frequent pattern when constructing a condensed FP-base?* In this section we will explore this question by providing a type of condensed FP-base and an efficient mining technique to find such a base.

Intuitively, we are going to construct a condensed FP-base consisting of maximal frequent patterns for a series of support thresholds. More specifically, given a support threshold min_sup and error bound k, we divide the set of frequent patterns into a number of disjoint subsets: (1) the set of patterns with support in the range $[min\_sup, min\_sup+k]$, (2) those with support in the range $[min\_sup+k+1, min\_sup + 2k + 1]$, and so on. The i-th subset contains those patterns with support in the range $[min\_sup + (i-1)(k+1), min\_sup + ik + i - 1]$, where $1 \leq i \leq \frac{|TDB|+1-min\_sup}{k+1}$. Given a frequent pattern, we can approximate its support with a maximal error of k, by determining which subset the pattern belongs to. To determine which subset a pattern belongs to, we only need to record the max-patterns at various layers w.r.t. the lower bounds of the supports of the ranges. The idea is illustrated in the following example.

Example 3. Given the transaction database TDB in Table 1, a support threshold of 1, and an error bound of 2, a condensed FP-base $\mathcal{B}_m$ can be constructed as follows.

Since the support threshold is 1 and the total number of transactions in the database is 7, we consider three ranges of supports: [1, 3], [4, 6], and [7, 7]. We mine max-patterns w.r.t. support thresholds 1, 4, and 7, respectively. The only max-pattern w.r.t. support threshold 1 is $abcd$, the max-patterns w.r.t. support threshold 4 are ab, c, and d, while there is no max-pattern w.r.t. support threshold 7. These four patterns form a condensed FP-base $\mathcal{B}_m$.

The base $\mathcal{B}_m$ is shown in Fig. 1c. The approximation function is $f_{\mathcal{B}_m}$, as defined in Example 1. In essence, for each given pattern X we find the super-pattern Y of X in $\mathcal{B}_m$ having the largest support, and use the range of the support for Y as the estimate of the support of X.

We now generalise the ideas by providing the definition of a condensed FP-base.

Definition 2. Given a transaction database TDB, a support threshold min_sup, and an error bound k, let the *number of levels* be

$$n\_level = \lfloor \frac{|TDB| + 1 - min\_sup}{k+1} \rfloor .$$

Define

$$\begin{aligned} min\_sup_1 &= min\_sup\,, \\ min\_sup_2 &= min\_sup + k + 1\,, \\ &\cdots \\ min\_sup_i &= min\_sup + (i-1)(k+1) \text{ for } (1 \leq i \leq n\_level)\,. \end{aligned}$$

Then, $\mathcal{B} = \bigcup_{i=0}^{i<n\_level} \mathcal{M}_i$ is called an *M-base* w.r.t. the approximation function ζ defined below. Here, $\mathcal{M}_i$ is the set of max-patterns w.r.t. support threshold min_sup_i.

The name *M-base* is used because the base is based on max-patterns.

Definition 3. Given an error bound k, an M-base $\mathcal{B}$, and a pattern X, let

$$\zeta(X) = \begin{cases} 0 & \text{if there exists no } X' \in \mathcal{B} \text{ s.t. } X' \supseteq X, \\ [sup(X), sup(X)] & \text{if } X \in \mathcal{B}, \\ [m, m+k] & \text{if } X \notin \mathcal{B}, \text{ where} \\ & m = \max\{sup(X') \mid X' \in \mathcal{B}, X' \supset X\}. \end{cases}$$

The following result shows that each M-base is not only a proper condensed FP-base w.r.t. function ζ, but also a minimal one.

Theorem 4.1. Each M-base $\mathcal{B}$ is a minimal condensed FP-base w.r.t. approximation function ζ.

Proof. We firstly show that $\mathcal{B}$ is a condensed FP-base w.r.t. ζ. For each pattern X, three cases arise.

- *Case 1:* $sup(X) < min\_sup$. Then there exists no max-pattern $Y \in \mathcal{B}$ such that $X \subseteq Y$. Thus, $\zeta(X) = 0$, and ζ correctly identifies X as an infrequent pattern.
- *Case 2:* $(sup(X) \geq min\_sup) \wedge (X \in \mathcal{B})$. Then $\zeta(X) = [sup(X), sup(X)]$, and so ζ approximates X's support without error.
- *Case 3:* $(sup(X) \geq min\_sup) \wedge (X \notin \mathcal{B})$. There must be some j $(1 \leq j \leq n\_level)$ such that $j = \max\{i \mid sup(X) \geq min\_sup_i\}$. Therefore, there exists some max-pattern $Y \in \mathcal{M}_j \subseteq \mathcal{B}$ w.r.t. support threshold min_sup_j s.t. $X \subseteq Y$. Let $Y_0 \in \mathcal{B}$ be the max-pattern with the largest support. Then, we have

  $$min\_sup_j \leq sup(Y_0) \leq sup(X) < min\_sup_{j+1} = min\_sup_j + k + 1$$

  for $(j < n\_level)$, or

  $$min\_sup_{n\_level} \leq sup(Y_0) \leq sup(X) \leq |TDB| \leq min\_sup_{n\_level} + k$$

  for $(j = n\_level)$. Hence, $\zeta(X)$ approximates $sup(X)$ with $[sup(Y_0), sup(Y_0) + k]$ (within the error bound k).

Hence, $\mathcal{B}$ is indeed a condensed FP-base w.r.t. ζ. To show that $\mathcal{B}$ is minimal, we only need to note that, for each pattern $X \in \mathcal{B}$, $sup(X)$ cannot be approximated properly by ζ using $\mathcal{B} - \{X\}$.

The remaining problem is *how to find the max-patterns efficiently in the condensed FP-base* $\mathcal{B}_m$.

There are many methods for mining max-patterns, such as MaxMiner (Bayardo 1998), Depth-first Search (Agarwal et al. 2001), MAFIA (Burdick et al. 2001), and GenMax (Gouda and Zaki 2001). A naïve method to compute $\mathcal{B}_m$ is to call a max-pattern mining algorithm multiple times, once for each lower bound of the ranges as a support threshold.

How do we mine the patterns of M-bases more efficiently than the naïve method?

Roughly speaking, we propose an algorithm to mine the database only once, for all the max-patterns w.r.t. the series of support thresholds. The algorithm proceeds in a depth-first manner. Moreover, our algorithm also uses other new pruning techniques. We will demonstrate the spirit of our algorithm with the following example.

Example 4. Consider the mining of max-patterns w.r.t. support thresholds 1 and 4 in the M-base $\mathcal{B}_m$ for the transaction database *TDB* of Table 1.

By scanning the transaction database *TDB* once, all frequent items, namely $a : 5$, $b : 5$, $c : 4$, and $d : 4$, are found. These items are sorted in support descending order, producing the list *F-list*$= a - b - c - d$.

F-list can be used to divide all max-patterns into four disjoint subsets: (1) the set of max-patterns containing item a; (2) those containing item b but not a; (3) those containing item c but not a nor b; and (4) those containing item d, i.e., the pattern d itself, if it is a max-pattern. We mine these four subsets of max-patterns one by one.

1. To find max-patterns containing item a, we form the *a-projected database* TDB_a by collecting all transactions containing item a, namely b, bc, bcd, and bd.
   Items b, c, and d are local frequent items in TDB_a. A list $F\text{-}list_a = b - c - d$ is formed by sorting these local frequent items in local support descending order. Based on $F\text{-}list_a$, all max-patterns containing item a can be further divided into four disjoint subsets: (1) pattern a itself, if it is a max-pattern; (2) the ones containing ab; (3) the ones containing item ac but not b; and (4) pattern ad if it is a max-pattern. We mine them one by one recursively.
   (a) The support of b in TDB_a is 4, denoted by $sup_{TDB_a}(b) = 4$. Since $sup(ab) = sup_{TDB_a}(b) = 4$, pattern a is not a max-pattern w.r.t. support threshold 4.
   (b) To find max-patterns containing ab, we form the *ab-projected database* TDB_{ab}, which contains c, cd, and d. Items a and b are omitted in TDB_{ab}, since they appear in every transaction in the ab-projected database. There is no item having support 4 or over in TDB_{ab}. Thus, ab is a max-pattern w.r.t. support threshold 4 (the lower bound of the second range of supports).
   Items c and d are frequent in TDB_{ab}. We recursively mine max-patterns by forming projected databases. It can be checked that $abcd$ is a max-pattern w.r.t. support threshold 1. Thus, the max-patterns containing ab are $ab : 4$ itself and $abcd : 1$.
   (c) To find max-patterns containing ac but not b, we form the ac-projected database TDB_{ac}, which contains d. Here, items a, b, and c are omitted since ac appears in every transaction and b occurs before c in *F-list*. The only frequent item in TDB_{ac} is d. However, $ac \subset abcd$ and $4 > sup(ac) > sup(abcd) = 1$. That means there exists no max-pattern containing ac but not b.
   (d) Since $4 > sup(ad) > sup(abcd) = 1$, ad is not a max-pattern.
   Therefore, the max-patterns containing a are ab and $abcd$.
2. To find all max-patterns containing b but not a, we form the b-projected database, which contains c, cd, d, and cd. The local frequent items in TDB_b are c and d, and $F\text{-}list_b = c - d$. The max-patterns containing b but not a can be divided into three subsets: (1) pattern b itself, if it is a max-pattern; (2) the ones containing bc; and (3) pattern bd, if it is a max-pattern. Let us mine them one by one.
   Since $b \subset ab$ and ab is a max-pattern, b is not a max-pattern.
   Since $sup(bc) = sup_{TDB_b}(c) = 2$, we have $4 > sup(bc) \geq sup(bcd) > sup(abcd)$. It follows that there are no max-pattern containing bc but not a.
   Similarly, we can check that bd is not a max-pattern.
   Thus, there are no max-patterns containing b but not a.
3. To mine max-patterns containing c but not a nor b, we can form the c-projected database and mine recursively. It can be verified that c is the only such max-pattern.
4. It can be verified that d is a max-pattern.

We have now found the complete set of max-patterns for condensed FP-base $\mathcal{B}_m = \{ab : 4, abcd : 1, c : 4, d : 4\}$.

As shown in the example, the general framework is in a depth-first manner. A list of frequent items in support descending order, called *F-list*, is used to divide the data as well as the mining task. In general, given *F-list*$= x_1 \cdots x_n$, the set of max-patterns can be divided into n disjoint subsets: the i-th subset contains the max-pattern having item x_i but none of x_j $(0 < j < i)$.

To mine max-patterns containing $X = x_{i_1} \cdots x_{i_m}$ (items in X are listed according to *F-list*), an *X-projected database* TDB_X is formed: every transaction $t = (tid, Y) \in TDB$ such that $X \subset Y$ is projected onto TDB_X as (tid, Y'); only items after x_{i_m} in the *F-list* are in Y'. In Example 4, *F-list*$= a - b - c - d$. Thus, the *ac-projected database* TDB_{ac} contains only one transaction, d (see step 1c). Here, the transaction-id is omitted.

The pruning techniques used in the mining are verified as follows.

Firstly, *how can we determine whether a frequent pattern X is a local max-pattern?* We have the following lemma.

Lemma 4.1. Let X be a frequent pattern and $i_X = \max\{i \mid sup(X) \geq min\_sup_i\}$. Then, X is a max-pattern w.r.t. $min\_sup_{i_X}$ if and only if X is not a sub-pattern of any max-pattern w.r.t. $min\_sup_{i_X}$ and $sup_{TDB_X}(x) < min\_sup_{i_X}$ for each item x in TDB_X.

Proof. (Direction IF) For each pattern $X = x_1 \cdots x_n$ (items in X are listed according to *F-list*) as stated in the lemma, let us consider $sup(Xy)$ for every $y \notin X$. Clearly, if y is after x_n in *F-list*, then $sup(Xy) < min\_sup_{i_X}$. In the case that y is before x_n in *F-list*, since X is not a sub-pattern of any max-pattern w.r.t. $min\_sup_{i_X}$, we have $sup(Xy) < min\_sup_{i_X}$. Thus, following the definition of "max-pattern", X is a max-pattern w.r.t. $min\_sup_{i_X}$.

(Direction ONLY-IF) The statement follows from the definition of "max-pattern" immediately.

In step 1b of Example 4, pattern *ab* is determined as a max-pattern w.r.t. support threshold 4 according to Lemma 4.1.

Secondly, *can we prune some unpromising patterns as early as possible?* We have the following lemma.

Lemma 4.2. Let X be a frequent pattern and *F-list*$_X = y_1 - \ldots - y_m$ be the *F-list* of local frequent items in TDB_X. For an item y_i in *F-list*$_X$, if there exists a max-pattern Z and i $(1 \leq i \leq n\_level)$ such that $(X \cup y_i \cdots y_m) \subseteq Z$ and

$$min\_sup_i \leq sup(Z) \leq sup_{TDB_X}(y_i) < min\_sup_{i+1},$$

then for $Y \subseteq y_i \cdots y_m$, $X \cup Y$ cannot be a max-pattern, and thus $(X \cup y_i)$-, ..., $(X \cup y_m)$-projected databases can be pruned.

Proof. We only need to notice the following two facts: For X and y_i as stated in the lemma, (1) $X \cup y_i \cdots y_m \subseteq Z$ is not a max-pattern (by Lemma 4.1). (2) From X and $y_i \cdots y_m$, we cannot derive any max-pattern which is a super-pattern of Z, since $X \cup y_i \cdots y_m \subseteq Z$. Thus, we have the lemma.

In step 2 of Example 4, we do not need to form and mine the *bc*-projected database since (1) the frequent items in the *b*-projected database are c and d with

support less than 4; and (2) *bcd* is not a max-pattern w.r.t. support threshold 1. Thus, Lemma 4.2 is applied here.

Based on the above analysis, we summarise the algorithm for constructing an M-base as follows.

Algorithm 2 (*CFP-M*: mining max-patterns at various layers).

Input: transaction database TDB, support threshold min_sup, and error bound $k\%$;

Output: an M-base $\mathcal{B}$ w.r.t. ζ;

Method:

let I be the set of all items; call $mine(TDB, \emptyset, I)$.

Function $mine(DB_X, X, I_X)$

// DB_X: a projected database, X: a frequent pattern, I_X: a set of items to be processed

1. scan DB_X once to find all frequent items within I_X;
2. let F_e be the set of items appearing in every transaction in DB_X, i.e., $F_e = \{x \mid x \in I_X, sup(x) = |DB_X|\}$; let $F_r = F_X - F_e$;
3. let $i = \max\{j \mid |DB_X| \geq min\_sup_j\}$;
   if $min\_sup_i > sup(y)$ for each item $y \in F_r$, and $X \cup F_e$ is not contained in any max-pattern w.r.t. support threshold min_sup_i, then output $X \cup F_e$;
   // $X \cup F_e$ is a max-pattern w.r.t. min_sup_i.
   // This step is based on Lemma 4.1.
4. let $F\text{-}list_X$ be the list of items in F_r in support descending order;
   for each item $x \in F\text{-}list_X$ (processed in the order) do
   (a) if the pruning criteria of Lemma 4.2 is satisfied for X, x (as y_i), and F_r (as $F\text{-list}_X$), then return;
   (b) otherwise, let $DB_{Xx} \subset DB$ be the subset of transactions containing x;
   let $I_{Xx} \subset F_r$ be the set of frequent items after x in F_X;
   call $mine(DB_{Xx}, F_e \cup \{x\}, I_{Xx})$;
5. return;

Analysis. The correctness of the algorithm follows from the lemmas given above. In this algorithm, we do not check every frequent pattern. Instead, we only check frequent patterns without a proper super-pattern that have exactly the same support count. Furthermore, by using Lemma 4.2, we prune patterns approximately contained by other max-patterns.

The implementation of Algorithm 2 involves projected databases and containment tests of frequent patterns. Accordingly, we propose the following two implementation optimisations.

- We use FP-tree (Han et al. 2000) to compress the database and projected databases. An FP-tree is a prefix tree storing transactions. Only frequent items in transactions are stored. From FP-trees, projected databases can be derived efficiently.
- One critical implementation issue of Algorithm 2 is that we need to identify max-patterns that contain a given pattern and stay in the same support range. In our implementation, we index max-patterns of the condensed FP-base by support level i (i.e., the pattern is w.r.t. min_sup_i) and length. Moreover, to facilitate the search, we organise all max-patterns by using a prefix tree, while all the nodes with the same item label are linked together.

5. Experimental Results and Performance Study

To evaluate the effectiveness and efficiency of condensed FP-bases, we conducted a comprehensive set of experiments. In this section, we report a summary of our results. All experiments were conducted on a PC with a Pentium III-750 CPU and 188 Mb main memory. All the programs were coded using Microsoft Visual C++6.0.

We used both synthetic datasets and real datasets in the experiments. The results are consistent. Due to lack of space, we only report results on three datasets as follows.

Firstly, to report results on the effectiveness and efficiency of FP-bases, we used the following dense datasets. A dataset is dense if it contains many long patterns even though the support threshold is relatively high. Mining frequent patterns from dense databases is very challenging.

- Real dataset *Mushroom*. The *Mushroom* dataset obtained from the UC-Irvine Machine Learning Database Repository has 8124 transactions, while the average length of a transaction is 23. It is a typical dense dataset.
- Real dataset *Connect-4*. This dataset was also obtained from the UC-Irvine Machine Learning Database Repository. It has 67 557 transactions and each transaction has 43 items. It is frequently considered a "difficult-to-mine" dense dataset.

To test the scalability of FP-bases, we also used a synthetic dataset $T10I4D100-1000k$. This dataset was generated by using the well-known IBM synthetic data generator (Agrawal and Srikant 1994). It is a sparse dataset simulating the market basket data. The number of transactions in this dataset is up to 1 million.

In our experiments, we compared the following three algorithms for mining condensed FP-bases.

- *CFP-D*: the level-by-level method for constructing condensed FP-base $\mathcal{B}_d$, i.e., Algorithm 1.
- *CFP-CLOSET*: we adapted the CLOSET algorithm (Pei et al. 2000) to *CFP-CLOSET* for mining condensed FP-base $\mathcal{B}_m$ as follows. CFP-CLOSET finds frequent closed patterns and checks whether a frequent closed pattern is in $\mathcal{B}_m$ according to Lemma 4.1. Only frequent closed patterns are output.
- *CFP-M*: it is Algorithm 2, which finds condensed FP-base $\mathcal{B}_m$ with all pruning and optimisation.

5.1. Effect of Compression

The compression effects of condensed FP-bases can be measured by the compression ratio defined in (1). Please note that the smaller the compression ratio, the better the compression effect.

Firstly, we fixed the support threshold and tested the compression ratio with respect to various error bounds. The results on datasets *Mushroom* and *Connect-4* are shown in Figs. 2 and 3, respectively.

Here, the error bound is set as a percentage of the total number of transactions in the dataset. If there are 1000 transactions in the dataset, then an error bound of 0.1% means that the absolute error bound is 1.

It is clearly shown that condensed FP-base $\mathcal{B}_m$ can achieve a much better compression ratio than $\mathcal{B}_d$. For example, in dataset *Mushroom*, when the support thresh-

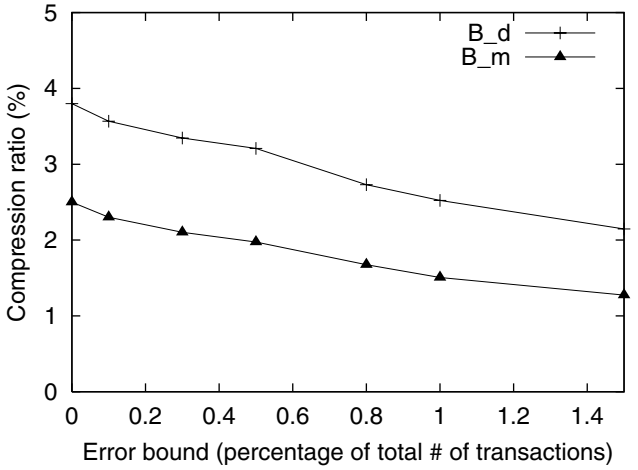


Fig. 2. The compression ratio of $\mathcal{B}_d$ and $\mathcal{B}_m$ on dataset *Mushroom* ($min\_sup = 14\%$).

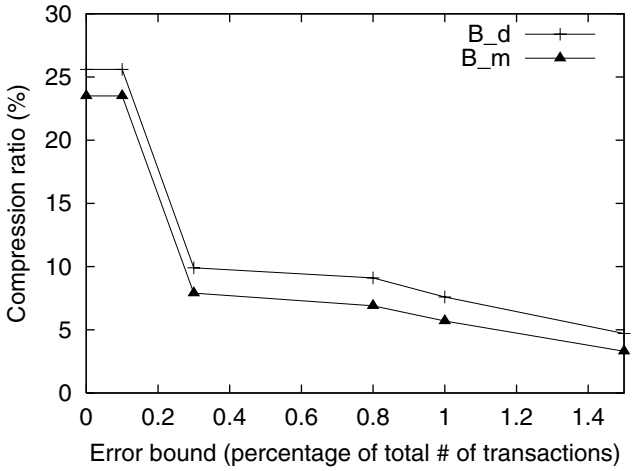


Fig. 3. The compression ratio of $\mathcal{B}_d$ and $\mathcal{B}_m$ on dataset *Connect-4* ($min\_sup = 93\%$).

old is set to 14%, there are in total 103 845 frequent patterns, and 2591 frequent closed patterns. As shown in Fig. 2, $\mathcal{B}_m$ is much smaller than $\mathcal{B}_d$. For both condensed FP-bases, the larger the error bound, the better the compression ratio.

Please note that, when the error bound is 0%, $\mathcal{B}_m$ is exactly the set of frequent closed patterns. As can be seen, frequent closed itemsets can achieve a good compression ratio. Condensed FP-base $\mathcal{B}_m$ can carry the benefit and take advantage of the error bound to do even better compression.

Since condensed FP-base $\mathcal{B}_m$ performs better than $\mathcal{B}_d$, we now focus on the compression effect of $\mathcal{B}_m$ with respect to support threshold. The results are shown in Figs. 4 and 5, respectively.

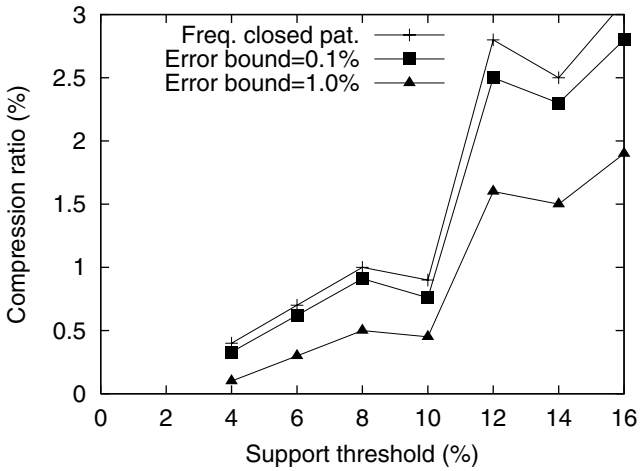


Fig. 4. The compression ratio of $\mathcal{B}_m$ w.r.t. support threshold on dataset *Mushroom*.

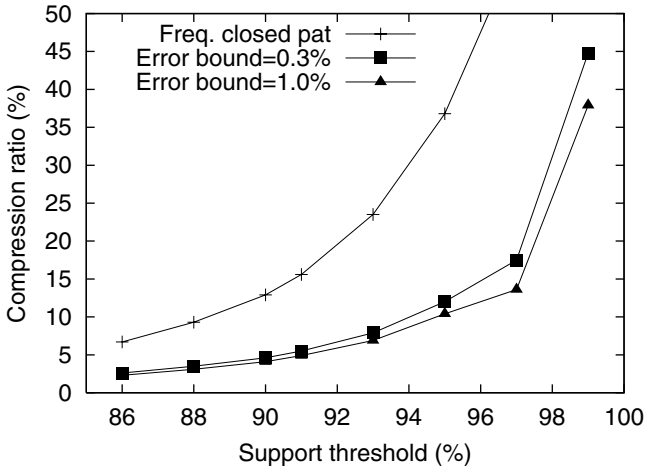


Fig. 5. The compression ratio of $\mathcal{B}_m$ w.r.t. support threshold on dataset *Connect-4*.

To help verify the compression effect, we also plot the compression ratio of the condensed FP-base using frequent closed patterns. A condensed FP-base using frequent closed patterns is one with an error bound of 0. As is clearly shown in the two figures, the larger the error bound, the better the compression. The results also confirm that, even with a small error bound, condensed FP-base $\mathcal{B}_m$ can be much smaller than the condensed FP-base of frequent closed patterns.

Moreover, the compression ratio is sensitive to the distribution of frequent patterns with respect to a specific support threshold. Fortunately, the general trend is that the lower the support threshold, the better the compression. When the support threshold is low, there are many frequent patterns with similar support counts. Thus, one pattern in a condensed FP-base may be a "representative" of many patterns.

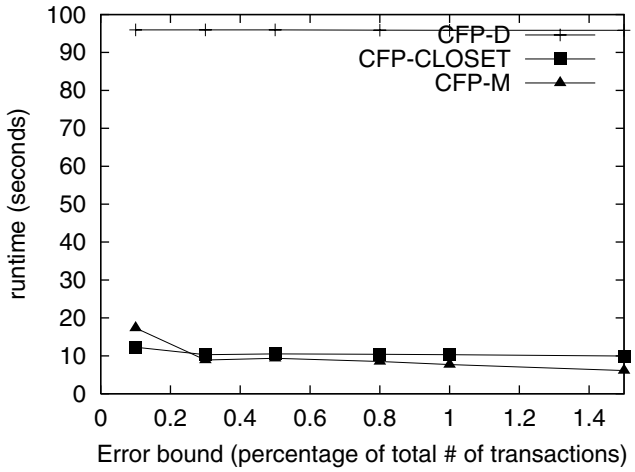


Fig. 6. The runtime w.r.t. error bound on dataset *Connect-4* ($min\_sup = 93\%$).

Similar trends can be observed for the compression effect of $\mathcal{B}_d$, but the compression ratio of $\mathcal{B}_d$ is larger than that of $\mathcal{B}_m$ in the same setting, i.e., the compression power of $\mathcal{B}_d$ is weaker. Limited by space, we omit the details here.

5.2. Efficiency of Computing Condensed FP-Bases

We compare the runtime of *CFP-D*, *CFP-CLOSET*, and *CFP-M* with respect to various error bounds in Fig. 6. The support threshold is set to 93%.

From the figure, we can see that the trends are as follows. The runtimes of both *CFP-D* and *CFP-CLOSET* is insensitive to the error bound. The two methods find the complete set of frequent patterns and frequent closed patterns, respectively, which are their dominant costs. We note that the cost of computing $X.ub$ for pattern X in *CFP-D* and that of the super-pattern checking in *CFP-CLOSET* are very minor compared with the expensive pattern mining in these two algorithms.

CFP-D fully utilises the error bound to prune the search space. The larger the bound, the faster the execution. Thus, it is faster than the other two algorithms when the error bound is not very small.

We observe a similar trend in dataset *Mushroom*. Limited by space, we omit the details here. Moreover, since *CFP-D* is dramatically slower than *CFP-CLOSET* and *CFP-M*, in the remainder of this subsection, our discussion focuses on *CFP-CLOSET* and *CFP-M*.

In Fig. 7, we compare the runtimes of *CFP-CLOSET* and *CFP-M* with respect to support threshold. The error bound was set to 0.1% of the total number of transactions in the dataset. When the support threshold is high, the runtimes of both methods are close. However, when the support threshold is low, the runtime of *CFP-CLOSET* increases dramatically, since it has to mine and check the complete set of frequent closed patterns. The runtime of *CFP-M* increases moderately even when the support threshold is low, since the pruning techniques help to confine the search in a small subset of frequent closed patterns.

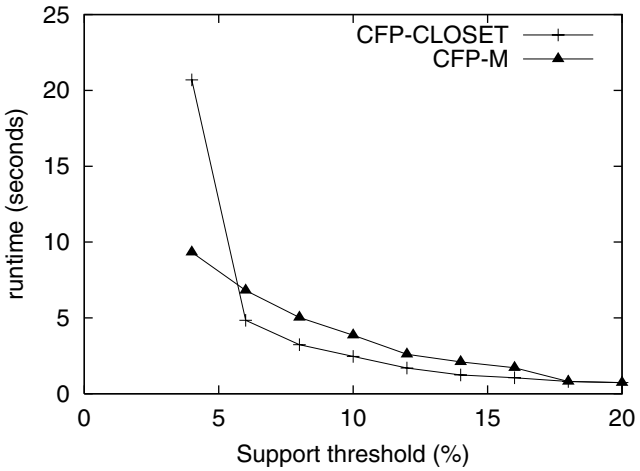


Fig. 7. The runtime w.r.t. support threshold on dataset *Mushroom* ($err\_b = 0.1\%$).

Again, similar trends are observed in experiments on other datasets. We omit the details here.

5.3. Scaling-up Test

We also tested the scalability of condensed FP-bases as well as related algorithms.

Firstly, we tested the scalability of the compression ratio of condensed FP-bases. (If the curve is flatter, we say that the curve is more scalable, since the compression ratio is not sensitive to the database size.) In Fig. 8, we show the results on dataset *Connect-4*. We fix the support threshold at 90% of the number of transactions in the tests, and vary the number of transactions from 10% to 100% of that in the original dataset. In the figure, we compare the compression ratio of an FP-base using frequent closed patterns with that of B_m. Interestingly, as the number of transactions increases, the compression ratio also increases. The reason is that, when there are more transactions, there are more patterns with different supports. Thus, the compression effect is not as good as that in the databases with small numbers of transactions. Fortunately, both the number of frequent closed patterns and that of patterns in B_m do not increase dramatically. Moreover, B_m is more scalable, since its compression ratio increases in a more moderate way.

Secondly, we used the synthetic dataset $T10I4D100 - 1000k$ to demonstrate the scalability of Algorithm *CFP-M*. To make a comparison with traditional frequent-pattern mining, we include the runtime of CLOSET in the figure. CLOSET computes the set of frequent closed patterns. The results are shown in Fig. 9. In this test, the error bound for *CFP-M* was set to 0.1%. From the figure, we can see that both methods are scalable with respect to the number of transactions in the datasets. Their runtimes are also close. CLOSET is faster when the database is large, since it does not need to check against the error bound. *CFP-M* has a scalability comparable to CLOSET and, at the same time, achieves non-trivial compression.

In summary, from the experimental results, we can draw the following conclusions. Firstly, condensed FP-bases can achieve non-trivial compression for frequent

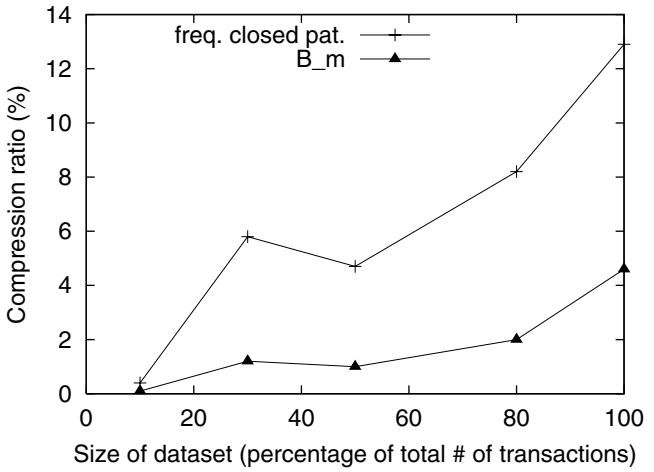


Fig. 8. Scalability of the compression ratio on dataset *Connect-4* ($min\_sup = 90\%$).

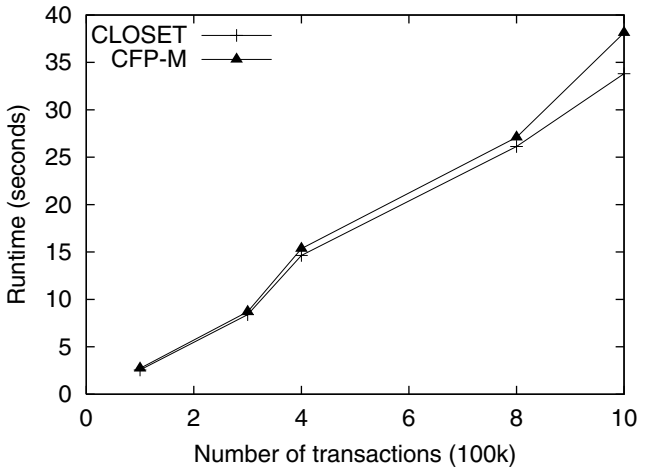


Fig. 9. Scalability of the runtime on dataset $T10I4D100 - 1000k$.

patterns. B_m often performs considerably better than B_d, and thus is more preferable. Secondly, the larger the error bound, the more we compress. The error bound can help to make the condensed FP-bases more compact. Thirdly, *CFP-M* is an efficient and scalable algorithm for computing condensed FP-base B_m. It is comparable to CLOSET in terms of runtime and scalability, and $\mathcal{B}_m$ achieves a better compression effect than the set of all frequent closed patterns. The optimisation and pruning techniques help make *CFP-M* efficient and scalable. Overall, B_m and *CFP-M* are the clear winners for frequent-pattern base compression and the corresponding computations.

6. Discussion

In this section, we firstly discuss the relationships between the complete FP-base and various condensed FP-bases. Then, we discuss some potential applications and extensions of condensed FP-bases.

6.1. What Kinds of Frequent Patterns Should a User Store?

Frequent-pattern mining approaches can be categorised into several themes, depending on the patterns mined: *all frequent patterns*, *frequent closed patterns*, *max-patterns*, and *condensed FP-bases*. *What kinds of frequent patterns should one mine and in what situation?* and *what are the corresponding benefits and overheads?* A concise comparison is presented as follows.

Mining all frequent patterns. The complete set of frequent patterns is the basis of all other kinds of "condensed sets" of frequent patterns, and the latter are concise representations of the former.

With the complete set of frequent patterns and the corresponding well-designed index, one can quickly determine whether a pattern is frequent and the support of a frequent pattern. However, the number of frequent patterns in a large database is often huge and computing the complete set of frequent patterns is often costly or even computationally prohibitive. An even more serious problem is that it is very hard or even impossible for users to comprehend a huge number of frequent patterns.

Mining frequent closed patterns. As shown before, a set of frequent closed patterns is a condensed FP-base. In many cases, the number of frequent closed patterns is much smaller than that of all frequent patterns. Thus, mining frequent closed patterns improves mining all frequent patterns in terms of effectiveness: a smaller set of patterns is easier to understand. Moreover, the exact support of a frequent pattern can be determined from a set of frequent closed patterns. To achieve this, we need to build a proper index and perform efficient containment checking.

Some algorithms have been developed and proposed to mine frequent closed patterns, such as CHARM (Zaki and Hsiao 2002) and CLOSET (Pei et al. 2000). Often, mining frequent closed patterns can be faster than mining all frequent patterns. However, when the database is sparse, mining frequent closed patterns can be more expensive than mining all frequent patterns. The major overhead is checking whether a frequent pattern is closed.

Mining max-patterns. Max-patterns (Bayardo 1998) are the borders of frequent patterns. The set of max-patterns can be much smaller than that of all patterns and that of frequent closed patterns. Given a set of max-patterns, one can determine whether a pattern is frequent but cannot determine the support count of a frequent pattern if it is not in the set.

Often, mining max-patterns is faster than mining all frequent patterns or frequent closed patterns.

Mining condensed FP-bases. Condensed FP-bases are more condensed representations than frequent closed patterns. With proper design, such as $\mathcal{B}_m$ in this paper, a condensed FP-base can be smaller than the set of frequent closed patterns. While achieving a better compression ratio than the set of frequent closed pat-

terns, a condensed FP-base can be used to determine the support of a frequent pattern approximately with a maximal error bound.
In particular, in $\mathcal{B}_m$, we chose a subset of closed patterns to form a condensed FP-base. Thus, the users are given the flexibility to balance the trade-off between the space overhead and the accuracy of the support count information.
Computing a condensed FP-base can be faster or slower than mining a set of frequent closed patterns. The major overhead of computing a condensed FP-base is to determine whether a frequent pattern should be materialised in the base.

As shown above, all frequent patterns, frequent closed patterns, max-patterns, and condensed FP-bases provide a full range of choice and flexibility. One advantage of condensed FP-bases is that the user can control the trade-off between base size and error bound.

6.2. Recovering the Complete Set of Frequent Patterns From a Condensed FP-Base

In some cases, it may be necessary to recover the complete set of frequent patterns. In this subsection, we present an efficient method of fulfilling the requirement on an M-base. The complete set of frequent patterns can be derived similarly from a level-by-level condensed FP-base $\mathcal{B}_d$.

Let X and Y be two patterns in an M-base $\mathcal{B}$ such that $X \subset Y$. X is called a *parent*[2] of Y in $\mathcal{B}$, denoted by $X \prec Y$, if there exists no pattern $Z \in \mathcal{B}$ such that $X \subset Z \subset Y$.

Clearly, the relation $\prec$ is a strict partial order over the patterns in $\mathcal{B}$. If we build an index on patterns in $\mathcal{B}$ such that (1) patterns are sorted in support descending order; and (2) the link between a pattern and all of its parents are recorded, then the complete set of frequent patterns can be derived by scanning M-base $\mathcal{B}$ only once as follows.

We scan the patterns in the order of $\prec$. For each pattern X, there are two cases:

- *X has no parent in $\mathcal{B}$*. We generate all non-empty proper subsets of X, if X has not been generated before. Their supports can be approximated as $[sup(X), sup(X) + k]$, where k is the error bound.
- *X has parents $Y_1, \ldots, Y_n$*. We generate all proper supersets Z of X such that $X \subset Z \subset Y_i$ $(1 \leq i \leq n)$, if Z has not been generated before. Clearly, Z's can be generated systematically by the differential of X and Y_i. For each Z, its support can be estimated as $[sup(X), sup(X) + k]$.

It can be verified that the above process generates the complete set of frequent patterns with proper approximated supports.

In order to derive the exact support of the frequent patterns, one can scan the database once and only count those patterns with approximate supports; i.e., the patterns in the condensed FP-base do not need to be counted again.

6.3. Applications and Extensions of Condensed FP-Bases

Condensed FP-bases can be used to solve many other data-mining problems. For example, one can use condensed FP-bases to build Bayesian belief networks, like

[2] We call X a parent of Y since X is above Y in the lattice graph analogous to the graphs in Fig. 1.

the approach in Margaritis et al. (2001). Since a condensed FP-base contains a much smaller number of frequent patterns, working with condensed FP-bases may be more efficient than working with all frequent patterns. We illustrate the potential applications in the following two examples.

Case 1. Let us consider mining frequent patterns from market basket data. The mining results may be used to generate association rules, build a classifier for customers, build a product recommender, and cluster customers based on their frequent purchasing patterns. As all these tasks need information about frequent patterns, to avoid duplicated mining of the patterns from the database, we should mine them once and store them. A complete set of frequent patterns could be huge. That brings non-trivial challenges for storage and retrieval.

Mining condensed FP-bases may solve the problem nicely, if an approximation with a user-specified error bound is acceptable. Firstly, a condensed FP-base is mined with the minimum error bound acceptable to all tasks. Then, association rules, the classifier, the product recommender, and the clusters of customers can be derived from the condensed FP-base. Since the condensed FP-base may be substantially smaller than the complete set of frequent patterns, the mining is more efficient while at the same time the quality of the mining results is guaranteed.

Case 2. Mining condensed FP-bases is a promising approach for mining frequent patterns in transaction streams. When mining transaction streams, it is very likely that the system does not have enough resources (including time and storage space) to maintain the complete set of frequent patterns. Thus, it is natural to introduce approximations. Condensed FP-bases can be used as synopses of transaction streams. Developing techniques of mining and maintaining a condensed FP-base is an interesting and important research problem.

It is also interesting to explore other approaches to construct condensed FP-bases. For example, we can consider using semantic relationships between patterns in defining condensed FP-bases, instead of simply using syntactic relationships. When we consider semantic relationships, we associate with each frequent pattern X the set $SAT(X)$, which is defined as the set of all transactions t in the database such that $X \subseteq t$. Possible semantic relationships include containment ($SAT(X) \subseteq SAT(Y)$) or large overlap, where $SAT(X) \cap SAT(Y)$ is relatively large compared with $SAT(X) \cup SAT(Y)$.

Another possible way to define condensed FP-bases is to allow statistically guaranteed error bounds of the estimated frequencies of patterns, instead of absolutely guaranteed error bounds as we have done in this paper. While this may lead to the loss of some frequency information, we conjecture that it may lead to much smaller bases.

7. Conclusions

In this paper, we introduced and considered the problem of mining a condensed frequent-pattern base. The notion of a condensed FP-base was introduced to significantly reduce the set of patterns that need to be mined, stored, and analysed, while providing a guaranteed error bound for frequencies of patterns not in the bases. We considered two types of condensed FP-bases: the downward condensed FP-base $\mathcal{B}_d$ and the max-pattern-based condensed FP-base $\mathcal{B}_m$. For the downward condensed FP-base, we used an a priori-style algorithm to mine it. For the max-pattern-based condensed FP-base, we introduced an interesting algorithm, together with several

novel optimisation techniques. Experimental results show that we can achieve a substantial compression ratio for condensation using condensed FP-bases, and that our algorithms are efficient and scalable. We also discussed some interesting extensions of our methods.

Acknowledgements. We are grateful to the anonymous reviewers of our ICDM'02 and KAIS submissions for their constructive comments, which helped improve the quality of the paper significantly. In particular, we thank the reviewers for informing us of some important related studies, and their suggestions for the enhancements in the motivation and applications. The work was supported in part by National Science Foundation under Grand No. 0209199 and 0308001, and the University of Illinois. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and not necessarily reflect the views of the funding agencies.

References

Agarwal R, Aggarwal C, Prasad VVV (2001) A tree projection algorithm for generation of frequent itemsets. J Parallel Distrib Comput 61:350–371

Agrawal R, Srikant R (1994) Fast algorithms for mining association rules. In: Proc 1994 Int Conf Very Large Data Bases (VLDB'94), Santiago, Chile, pp 487–499

Agrawal R, Srikant R (1995) Mining sequential patterns. In: Proc 1995 Int Conf Data Engineering (ICDE'95), Taipei, Taiwan, pp 3–14

Bayardo RJ (1998) Efficiently mining long patterns from databases. In: Proc 1998 ACM-SIGMOD Int Conf Management of Data (SIGMOD'98), Seattle, USA, pp 85–93

Beil F, Ester M, Xu X (2002) Frequent term-based text clustering. In: Proc 2002 ACM SIGKDD Int Conf Knowledge Discovery in Databases (KDD'02), Edmonton, Canada, pp 436–442

Beyer K, Ramakrishnan R (1999) Bottom-up computation of sparse and iceberg cubes. In: Proc 1999 ACM-SIGMOD Int Conf Management of Data (SIGMOD'99), Philadelphia, USA, pp 359–370

Boulicaut JF, Bykowski A, Rigotti C (2000) Approximation of frequency queries by means of free-sets. In: Principles of Data Mining and Knowledge Discovery, pp 75–85

Brin S, Motwani R, Silverstein C (1997) Beyond market basket: generalizing association rules to correlations. In: Proc 1997 ACM-SIGMOD Int Conf Management of Data (SIGMOD'97), Tucson, USA, pp 265–276

Burdick D, Calimlim M, Gehrke J (2001) MAFIA: a maximal frequent itemset algorithm for transactional databases. In: Proc 2001 Int Conf Data Engineering (ICDE'01), Heidelberg, Germany, pp 443–452

Dong G, Han J, Lam J, Pei J, Wang K (2001) Mining multi-dimensional constrained gradients in data cubes. In: Proc 2001 Int Conf Very Large Data Bases (VLDB'01), Rome, Italy, pp 321–330

Dong G, Li J (1999) Efficient mining of emerging patterns: discovering trends and differences. In: Proc 1999 Int Conf Knowledge Discovery and Data Mining (KDD'99), San Diego, USA, pp 43–52

Gouda K, Zaki MJ (2001) Efficiently mining maximal frequent itemsets. In: ICDM, pp 163–170

Han J, Dong G, Yin Y (1999) Efficient mining of partial periodic patterns in time series database. In: Proc 1999 Int Conf Data Engineering (ICDE'99), Sydney, Australia, pp 106–115

Han J, Pei J, Yin Y (2000) Mining frequent patterns without candidate generation. In: Proc 2000 ACM-SIGMOD Int Conf Management of Data (SIGMOD'00), Dallas, USA, pp 1–12

Imielinski T, Khachiyan L, Abdulghani A (2002) Cubegrades: generalizing association rules. Data Min Knowl Discovery 6:219–258

Kamber M, Han J, Chiang JY (1997) Metarule-guided mining of multi-dimensional association rules using data cubes. In: Proc 1997 Int Conf Knowledge Discovery and Data Mining (KDD'97), Newport Beach, USA, pp 207–210

Liu B, Hsu W, Ma Y (1998) Integrating classification and association rule mining. In: Proc 1998 Int Conf Knowledge Discovery and Data Mining (KDD'98), New York, USA, pp 80–86

Li W, Han J, Pei J (2001) CMAR: accurate and efficient classification based on multiple class-association rules. In: Proc 2001 Int Conf Data Mining (ICDM'01), San Jose, USA, pp 369–376

Lakshmanan LVS, Ng R, Han J, and Pang A (1999) Optimization of constrained frequent set queries with 2-variable constraints. In: Proc 1999 ACM-SIGMOD Int Conf Management of Data (SIGMOD'99), Philadelphia, USA, pp 157–168

Lent B, Swami A, Widom J (1997) Clustering association rules. In: Proc 1997 Int Conf Data Engineering (ICDE'97), Birmingham, England, pp 220–231,

Mannila H, Toivonen H (1996) Multiple uses of frequent sets and condensed representations (extended abstract). In: Knowledge Discovery and Data Mining, pp 189–194
Mannila H, Toivonen H, Verkamo AI (1997) Discovery of frequent episodes in event sequences. Data Min Knowl Discovery 1:259–289
Margaritis D, Faloutsos C, Thrun S (2001) Netcube: a scalable tool for fast data mining and compression. In: Proc 2001 Int Conf Very Large Data Bases (VLDB'01), pp 311–320
Ng R, Lakshmanan LVS, Han J, Pang A (1998) Exploratory mining and pruning optimizations of constrained associations rules. In: Proc 1998 ACM-SIGMOD Int Conf Management of Data (SIGMOD'98), Seattle, USA, pp 13–24
Pasquier N, Bastide Y, Taouil R, Lakhal L (1999) Discovering frequent closed itemsets for association rules. In: Proc 7th Int Conf Database Theory (ICDT'99), Jerusalem, Israel, pp 398–416
Pei J, Han J, Lakshmanan LVS (2001) Mining frequent itemsets with convertible constraints. In: Proc 2001 Int Conf Data Engineering (ICDE'01), Heidelberg, Germany, pp 433–332
Pei J, Han J, Mao R (2000) CLOSET: an efficient algorithm for mining frequent closed itemsets. In: Proc 2000 ACM-SIGMOD Int Workshop Data Mining and Knowledge Discovery (DMKD'00), Dallas, USA, pp 11–20
Pei J, Han J, Mortazavi-Asl B, Pinto H, Chen Q, Dayal U, Hsu M-C (2001) PrefixSpan: mining sequential patterns efficiently by prefix-projected pattern growth. In: Proc 2001 Int Conf Data Engineering (ICDE'01), Heidelberg, Germany, pp 215–224
Silverstein C, Brin S, Motwani R, Ullman J (1998) Scalable techniques for mining causal structures. In: Proc 1998 Int Conf Very Large Data Bases (VLDB'98), New York, USA, pp 594–605
Zaki M (2000) Generating non-redundant association rules. In: Proc 2000 ACM SIGKDD Int Conf Knowledge Discovery in Databases (KDD'00), Boston, USA, pp 34–43
Zaki MJ, Hsiao CJ (2002) CHARM: an efficient algorithm for closed itemset mining. In: Proc 2002 SIAM Int Conf Data Mining, Arlington, USA, pp 457–473

Author Biographies

Jian Pei received his B.Eng. and M.Eng. degrees, both in Computer Science, from Shanghai Jiaotong University, China, in 1991 and 1993, respectively, and his Ph.D. degree in Computing Science from Simon Fraser University, Canada, in 2002. He is currently an Assistant Professor of Computer Science and Engineering at the State University of New York at Buffalo, USA. His research interests include data mining, data warehousing, online analytical processing, database systems, and bio-informatics. He has published over 30 research papers in refereed journals, conferences, and workshops. He is a member of the editorial board of the ACM SIGMOD Digital Symposium Collection (DiSC). He has served on the program committees of international conferences and workshops, including KDD03, ICDM03, ICDE02, and CIKM02. He has frequently reviewed papers for leading academic journals, including ACM Transactions on Database Systems, IEEE Transactions on Knowledge and Data Engineering, Data Mining and Knowledge Discovery, and Knowledge and Information Systems. Dr Pei is a member of the ACM, the ACM SIGMOD, and the ACM SIGKDD.

Guozhu Dong received his B.S. degree in Mathematics from Shandong University, China, in 1982, and his M.S. and Ph.D. degrees from the University of Southern California, USA, in 1985 and 1988, respectively. Since 1999 he has been an associate professor at Wright State University, USA. Previously, he taught at the University of Melbourne and Flinders University, both in Australia. His research interests include databases, data mining, and bioinformatics. He has published over 80 articles in these areas. Dr Dong is a senior member of the IEEE and a member of the ACM.

Wei Zou received her Master's degree in Computer Software from Jiangxi Normal University in 1998 and her Ph.D. degree in Computer Science from Peking University in 2001. Her research activities are focused on datacube mining, text mining, and the indexing and retrieval of digital libraries. She is interested in all aspects of data mining and information retrieval.

Jiawei Han is a Professor in the Department of Computer Science at the University of Illinois at Urbana-Champaign. Previously, he was an Endowed University Professor at Simon Fraser University, Canada. He has been working on research into data mining, data warehousing, database systems, spatial databases, deductive and object-oriented databases, Web databases, bio-medical databases, etc., with over 150 journal and conference publications. He has chaired or served on many program committees of international conferences and workshops, including the 2001 and 2002 SIAM-Data Mining Conferences (PC co-chair), the 2002 International Conference on Data Engineering (PC vice-chair), the ACM SIGKDD conferences (2001 best paper award chair, 2002 student award chair), the 2002 and 2003 ACM SIGMOD conferences (2000 demo/exhibit program chair), etc. He has also served or is serving on the editorial boards for Data Mining and Knowledge Discovery: An International Journal, IEEE Transactions on Knowledge and Data Engineering, and the Journal of Intelligent Information Systems. He has also been serving on the Board of Directors of the Executive Committee of the ACM Special Interest Group on Knowledge Discovery and Data Mining (SIGKDD). His textbook *Data Mining: Concepts and Techniques* (Morgan Kaufmann, 2001) has been popularly used for data mining courses in many universities.

Correspondence and offprint requests to: Jian Pei, Department of Computer Science and Engineering, State University of New York at Buffalo, 201 Bell Hall, Buffalo, NY 14260-2000, USA.
Email: jianpei@cse.buffalo.edu