Knowl Inf Syst (2011) 29:55–80
DOI 10.1007/s10115-010-0329-5

REGULAR PAPER

# Efficient mining of all margin-closed itemsets with applications in temporal knowledge discovery and classification by compression

**Fabian Moerchen · Michael Thies · Alfred Ultsch**

Received: 26 September 2009 / Revised: 5 June 2010 / Accepted: 8 July 2010 /
Published online: 22 July 2010


**Abstract** Margin-closed itemsets have previously been proposed as a subset of the closed itemsets with a minimum margin constraint on the difference in support to supersets. The constraint reduces redundancy in the set of reported patterns favoring longer, more specific patterns. A variety of patterns ranging from rare specific itemsets to frequent general itemsets is reported to support exploratory data analysis and understandable classification models. We present DCI_MARGIN, a new efficient algorithm that mines the complete set of margin-closed itemsets. We modified the DCI_CLOSED algorithm that has low memory requirements and can be parallelized. The margin constraint is checked on-the-fly reusing information already computed by DCI_CLOSED. We thoroughly analyzed the behavior on many datasets and show how other data mining algorithms can benefit from the redundancy reduction.

**Keywords** Closed itemsets · Constrained itemsets · Condensed representation · Temporal data mining · Compression

## 1 Introduction

Itemset mining has long advanced over the initial concept of market basket analysis [2] and is used to tackle many data mining problems such as frequent pattern mining, association rule generation [36], clustering [5,26,49,74], classification [19,29,42,43,55,71,73,80] and temporal data mining [3,52]. The mining of itemsets is a core step in these methods that often dominates the overall complexity of the problem. The mining of frequent itemsets is a challenging task because the possible number of patterns can be extremely large even for

F. Moerchen (✉)
Siemens Corporate Research, 755 College Road East,
Princeton, NJ 08540, USA
e-mail: fabian.moerchen@siemens.com

M. Thies · A. Ultsch
Databionic Research Group, Philipps-University Marburg,
Hans-Meerwein-Str, 35032 Marburg, Germany

moderately sized datasets complicating a manual analysis or further automated processing steps [76].

Researchers have proposed many solutions to reduce the number of patterns depending on the context in which the patterns are used, for example, condensed representations [18], constrained itemsets [58] and combinations thereof [8,24]. For association rule generation, closed itemsets [56,10] are commonly used to avoid redundant rules [81] favoring longer patterns to generate specific rules. For frequency queries non-derivable itemsets [16] provide a compact lossless representation favoring shorter patterns to keep the summary small. Margin-closed itemsets have been previously proposed by the authors for exploratory knowledge discovery tasks in the context of temporal data mining [51,52] and independently as $\delta$-tolerance itemsets for frequency estimation in [22]. Margin-closed patterns are a specialization of closed itemsets with a constraint to limit the redundancy among reported patterns. An itemset is closed if no superset with the same frequency exists. An itemset is margin-closed if no superset with almost the same frequency exists, where 'almost' is defined by a threshold $\alpha$ on the relative (or absolute) difference of the frequencies. The threshold ensures a frequency *margin* among the reported patterns.

Note that margin-closed itemsets are *not* an error-tolerant approach and *not* an approximation to closedness. In contrast to pattern summarization, error-tolerant, and approximation only actually observed itemsets with their exact frequencies are reported. In contrast to non-derivable itemsets, the goal is not to support frequency queries with a compact summary but to provide long patterns with low redundancy to support exploratory data analysis tasks. Frequent patterns (per class) can help a human analyst understand the structure of a dataset. Less patterns with less redundancy are easier to comprehend. The bias toward longer patterns provides more explanation for each pattern to the user. When used as features in a classification model removing redundancy translates to faster training times and more concise models. In [22], the authors explore margin-closed itemsets as a condensed representation for frequency estimation. The frequency of non-margin-closed itemsets is approximated by the average frequency of the items in the common superset, again motivating the need for favoring longer itemsets.

In this paper, we study the problem of *efficiently mining of all frequent margin-closed itemsets from a database of itemset transactions*. The margin-closed itemsets are a subset of all closed itemsets and a superset of all maximal itemsets and the mining of closed and maximal itemsets has been well studied. The naive approach would be to mine all closed itemsets and check the margin constraint for each one. This can be done by comparing the support of a closed itemset to the support of all extensions with one additional item. Obviously this is computationally expensive in particular for low minimum support values that generate large numbers of closed itemsets. Incorporating the pruning from closed to margin-closed itemsets into the mining algorithms can be expected to be more efficient. We propose DCI_MARGIN, a new algorithm based on DCI_CLOSED [46] that efficiently mines all margin-closed itemsets. Several pruning techniques are introduced and the correctness and completeness of the algorithm is shown. Our solution checks the margin constraint on-the-fly reusing information already computed by DCI_CLOSED. Previous work has adapted the FP-Growth [34] and CHARM [82] algorithms for closed itemset mining. The former does not guarantee completeness due to greedy pruning heuristics [22], and the latter is less efficient due to required subsumption check [51]. We show that DCI_MARGIN can significantly reduce the number of reported patterns if there is redundancy with comparable or faster run time. The discovered redundancy at various minimum margin and support thresholds provides interesting insights into datasets from different domains. Our main contributions are:

- The efficient DCI_MARGIN algorithm that mines the complete set of margin-closed itemsets. Previous work used a variation of CHARM that has much higher memory requirements [51] and an adaption of FP-Growth [22] that used greedy pruning heuristics leading to incomplete results as demonstrated by our experiments.
- A thorough evaluation of both the pattern class and the algorithm with 60 datasets from various domains. Previous work has concentrated on the special case of temporal data mining [51] or used only few itemset datasets [22].
- A discussion of different applications of margin-closed itemsets.

In addition, we provide several examples of data mining applications using margin-closed patterns.

In the remainder of this paper, we motivate and define margin-closed itemsets in Sect. 2 and describe an efficient algorithm for their discovery in Sect. 3. Section 4 demonstrates how we can reduce the number of reported itemsets significantly and efficiently. Applications are described in Sect. 5. The results and related work are discussed in Sections 6-7.

## 2 Margin-closed itemsets

### 2.1 Basic notation

Given a finite set of items $\mathcal{I}$ and a finite set of transactions $I \subseteq \mathcal{I}$, represented by unique identifiers $\mathcal{T}$, a dataset can be described as the relation $\mathcal{D} \subseteq \mathcal{I} \times \mathcal{T}$. The function $g(I) = \{t \in T | \forall i \in I : (i,t) \in \mathcal{D}\}$ returns the transactions in which all items of itemset $I$ are included. The function $f(T) = \{i \in \mathcal{I} | \forall t \in T : (i,t) \in \mathcal{D}\}$ returns all items that are present in all transactions of $T$. The composite function $c = f \circ g$ is a closure operator (e.g., [46]). Let the support of an itemset $I$ be the fraction of transactions in which the itemset occurs: $supp(I) = \frac{|g(I)|}{|\mathcal{T}|}$.

**Definition 2.1** An itemset $I$ is called frequent w.r.t. a minimum support threshold $1 \geq \theta \geq 0$, if $supp(I) \geq \theta$. Let the set of all frequent itemsets be `FI`.

**Definition 2.2** An itemset $I \in$ `FI` is maximal if and only if $\forall I \subset I' \Rightarrow supp(I') < \theta$, i.e., if there is no frequent superset. Let the set of all maximal frequent itemsets be `MFI`.

**Definition 2.3** An itemset $I \in$ `FI` is (frequent-)closed, if and only if $\forall I' \subseteq \mathcal{I} : I \subset I' \Rightarrow supp(I') \neq supp(I)$, i.e., if there is no superset with the same support. Let the set of all closed frequent itemsets be `CFI`.

The closure operator $c$ partitions the lattice of the power sets of $\mathcal{I}$ into equivalence classes regarding the support. The unique suprema regarding the subset relation of those classes are the closed itemsets.

**Definition 2.4** An itemset $G \subseteq \mathcal{I}$ with $G = C \cup \{i\}$ [1] and $C \in$ `CFI` and $i \in \mathcal{I} \setminus C$ is called a *generator*.
Note, that this definition of a generator does not require minimality. A generator represents a seed itemset obtained by extending a closed itemsets with a new item, thus entering a new equivalence class. A generator can be used to obtain the closed itemset representing the class by adding all items that do not decrease the support.

[1] For the sake of brevity we write $C \cup i$ for $C \cup \{i\}$.

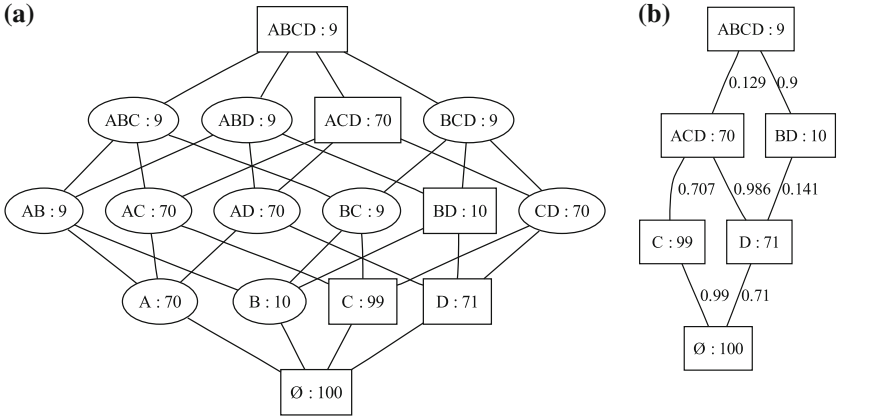



**Fig. 1** Itemset lattices for data from Table 1 and $\theta = 0.09$: **a** Frequent itemsets annotated with absolute support. **b** Closed itemsets and edges annotated with $\frac{supp(I')}{supp(I)}$ for $I \subset I'$

**Table 1** Example data: four itemsets with the number of transactions composed of exactly these items

| Itemset | $BD$ | $ABCD$ | $ACD$ | $C$ | (Total) |
|---|---|---|---|---|---|
| Transactions | 1 | 9 | 61 | 29 | 100 |

## 2.2 Margin-closedness

**Definition 2.5** An itemset $I \in \texttt{FI}$ is margin-closed w.r.t. a threshold $\alpha \in [0, 1]$ if and only if $\forall I' \in \texttt{FI} : I \subset I' \Rightarrow \frac{supp(I')}{supp(I)} < 1 - \alpha$, i.e., if there is no superset with *almost* the same support. Let the set of all margin-closed frequent itemsets w.r.t. to a threshold $\alpha$ be $\texttt{CFI}^{\alpha}$.

*Example 2.6* Consider the example database in Table 1. There are four itemsets composed of the four items $A$, $B$, $C$, and $D$. For each itemset the dataset contains the indicated number of transactions with exactly these itemsets. Figure 1a shows the lattice of all frequent itemsets for $\theta = 0.09$ with itemsets connected by the subset relation. The empty set at the bottom is present in all 100 transactions, the set of all items at the top has an absolute support of 9. The rectangles indicate closed itemsets. No items can be added to these sets without decreasing the support. Figure 1b shows only the closed itemsets and the subset relations annotated with $\frac{supp(I')}{supp(I)}$ for $I \subset I'$. If we set $\alpha = 0.1$, i.e., we require a relative support margin of at least 10%, the itemsets $\emptyset$, $D$ and $BD$ would not be considered margin-closed and removed from Fig. 1b. For example, the superset $ACD$ of $D$ has 98.6% of the support of $D$, so $D$ is considered redundant. In contrast, if the minimum support was set to $\theta = 0.1$ the itemset $BD$ would be margin-closed because the superset $ABCD$ would not be frequent anymore, making $BD$ a maximal itemset.

**Corollary 2.7** $\texttt{CFI}^{\alpha} \subseteq \texttt{CFI}$, *i.e., margin-closed frequent itemsets are a subset of the closed frequent itemsets.*

*Proof* Let $I \in \texttt{CFI}^{\alpha}$, then $\forall I' \in \texttt{FI}$ with $I \subset I'$: $\frac{supp(I')}{supp(I)} < 1 - \alpha \Rightarrow \text{supp}(I') < (1 - \alpha)\text{supp}(I) \leq \text{supp}(I) \Rightarrow \text{supp}(I') < \text{supp}(I) \Rightarrow I \in \texttt{CFI}$

For $\alpha > 0$ the margin-closedness condition is stricter than the closedness condition. We ensure a margin of support between a reported itemset and any frequent superset. For $\alpha = 0$ the set of margin-closed itemsets is equal to the set of all closed itemsets.

**Corollary 2.8** $\texttt{MFI} \subseteq \texttt{CFI}^{\alpha}$*, i.e., maximal frequent itemsets are a subset of the margin-closed itemsets.*

*Proof* Let $I \in \texttt{MFI} \Rightarrow \nexists I' \in \texttt{FI}$ with $I \subset I' \Rightarrow \nexists I' \in \texttt{FI}$ with $\frac{supp(I')}{supp(I)} < 1 - \alpha$ and $I \subset I' \Rightarrow I \in \texttt{CFI}^{\alpha}$

The margin-closedness condition is surely met if there are no frequent supersets at all, as is the case for maximal itemsets. For $\alpha = 1$ the set of margin-closed itemsets is equal to the set of maximal frequent itemsets, since the margin extends all the way to a support of 0. The equality might also hold for some values $\alpha < 1$ depending on the minimum support $\theta$ and the particular item frequencies in the data. If the relative minimum support threshold $\frac{\theta}{supp(\emptyset)}$ exceeds the margin threshold $1 - \alpha$, the margin condition can only be met by maximal frequent itemsets due to the following inequality for a frequent itemset $I$: $\theta \leq supp(I) = \frac{supp(I)}{supp(\emptyset)} < (1 - \alpha) * supp(\emptyset)$.

The threshold $\alpha$ of margin-closedness prunes itemsets with very similar support to a superset. Our reasoning behind this is that larger itemsets are more specific descriptions of patterns but that patterns that occur in almost the same transactions are redundant. The number of reported patterns is thus reduced without having to raise the minimum support threshold or retreating to maximal frequent itemsets keeping a variety of frequent general and rare specific patterns.

We want to emphasize that the goal is not to approximate the frequency of patterns that are not reported, though this might be possible based on margin-closed itemsets [22], but rather to understand the structure of transaction datasets. Approaches that aim at frequency estimation usually favor shorter patterns to achieve higher compression ratios.

## 3 Mining margin-closed itemsets

In this section, we describe our proposed DCI_MARGIN algorithm that modifies DCI_CLOSED to only report margin-closed itemsets given a threshold $\alpha$. We first describe the post-processing Algorithm 1 that can be combined with any algorithm for closed itemsets mining. It demonstrates the basic principle used to determine the margin-closedness of a closed itemset. Then, we integrate the margin-check into the variant of DCI_CLOSED algorithm for dense datasets and add several pruning steps to obtain the final Algorithm 2.

Algorithm 1 tests the margin condition for each closed frequent itemsets $C \in \texttt{CFI}$ using the TESTMARGIN function (Line 3). For all items $j \in \mathcal{I} \setminus C$ the support of the superset $C \cup j$ is calculated. If any superset is frequent and violates the margin condition (Line 8), the closed itemset is ignored, otherwise it can be reported as a margin-closed itemset (Line 12). Due to the monotonicity of support we only need to check supersets with one additional item.

Algorithm 2 is based on the DCI_CLOSED [46] algorithm that uses closure climbing and a vertical representation of the database. We first describe the inherited algorithmic steps briefly skipping the lines that deal with margin-closedness. The algorithm starts with $C$ initialized to the bottom closure, i.e., the set of items present in all transactions (possibly the empty set). $P$ initially contains all remaining frequent items and $D$ and $M$ are empty. The loop starting in Line 2 iterates over all items according to the total order $\prec$ (Line 3). Each item is removed from $P$ (Line 4) and added to the current closed itemset $C$ to obtain a closure generator $C_i$ of

an equivalence class. Line 6 checks whether the generator and thus the complete equivalence class is frequent and whether an equivalence class is entered that has already been visited by the algorithm (IsDuplicate). The duplicate check is performed by keeping track of two disjunct sets of items during the recursion ($D$ and $P$): those that would generate equivalence classes that were already visited and those that would generate previously unseen equivalence classes. We refer the interested reader to [46] for more explanations and proofs of the duplicate check and pruning technique.

**Algorithm 1** PostMargin
(Mining all margin-closed frequent itemsets from all closed frequent itemsets.)

1: **PROCEDURE** postMargin(`CFI` )
2: **for all** $C \in$ `CFI` **do**
3: TestMargin($C$);
4: **end for**
5: **END PROCEDURE**
6: **PROCEDURE** TestMargin($C$)
7: **for all** $j \in \mathcal{I} \setminus C$ **do**
8: **if** $supp(C \cup j) \geq \theta \wedge \frac{supp(C \cup j)}{supp(C)} \geq (1 - \alpha)$ **then**
9: **return**
10: **end if**
11: **end for**
12: **print** $C$;
13: **END PROCEDURE**

If the itemset $C_i$ passed the two tests in Line 6, the unique supremum $\hat{C}$ of the equivalence class generated by $C_i$ is found by adding all items that do not decrease the support (Line 13–14). All other items are collected in $P_i$ (Line 16) and passed in the recursive function call in Line 20, as those items that will create generators of new equivalence classes when added to $\hat{C}$. All items $i$ that were used to generate a new equivalence class are recorded in $D$, the so-called pre-set [46], to support the duplicate check (Line 24). All closed itemsets could be reported after line 18. All margin-closed itemsets could be reported by calling the TestMargin function of Algorithm 1. A better performance can be achieved by integrating the pruning strategies described below.

**Delay pruning** If we test the margin condition of closed itemsets immediately upon their discovery after Line 18 we have to generate the transaction list of $\hat{C} \cup j$ for all $j \in \mathcal{I} \setminus \hat{C}$. Some of these itemsets are also used in the recursive call in Line 20 to obtain generators of $C$ (instantiated with $\hat{C}$) in Line 5. By delaying the margin test until after the recursion (Line 22), we can utilize these results. We keep track of all items that generate supersets of $C$ which violate the margin condition in the set $M$ (Lines 7–9). When returning from the recursive call in Line 20, these items are available in $\hat{M}$. If any generator violated the margin condition (Line 21) we do not need to call TestMargin. Not all items in $P_i$ passed to the recursive call are tested in Line 7 because of the conditions in Line 6. A generator excluded by the first condition is infrequent and can therefore not violate the margin condition. A generator excluded by the second condition is a duplicate, i.e., the corresponding transaction list is a subset of the transaction list of a previously tested generator. If the previous test did not violate the margin condition, this generator does not need to be tested because it has a smaller or equal support. The margin test is thus performed for all necessary items in $P_i$ during the recursion. If $\hat{M}$ is empty after the recursion we still need to check the margin condition for all items in $D$ that generate equivalence classes derivable from the current closed itemset.

**Algorithm 2** DCIMARGIN
(Mining all margin-closed frequent itemsets.)

```
1: FUNCTION DCIMargin(C, D, P, M)
2: while P ≠ ∅ do
3:   i ← min≺(P);
4:   P ← P \ i;
5:   C_i ← C ∪ i;
6:   if (supp(C_i) ≥ θ) ∧ (¬IsDuplicate(C_i, D)) then
7:     if supp(C_i)/supp(C) ≥ 1 − α then
8:       M ← M ∪ i;
9:     end if
10:    Ĉ ← C_i;
11:    P_i ← ∅;
12:    for all j ∈ P do
13:      if g(C_i) ⊆ g(j) then
14:        Ĉ ← Ĉ ∪ j;
15:      else
16:        P_i ← P_i ∪ j;
17:      end if
18:    end for
19:    M̂ ← ∅;
20:    DCIMargin(Ĉ, D, P_i, M̂);
21:    if M̂ = ∅ then
22:      TestMargin(Ĉ, D);
23:    end if
24:    D ← D ∪ i;
25:  end if
26: end while
27: END PROCEDURE
28: FUNCTION IsDuplicate(C_i, D)
29: for all j ∈ D do
30:   if g(C_i) ⊆ g(j) then
31:     return true;
32:   end if
33: end for
34: return false;
35: END FUNCTION
36: PROCEDURE TestMargin(Ĉ, D)
37: for all j ∈ D do
38:   if supp(Ĉ ∪ j) ≥ θ ∧ supp(Ĉ ∪ j)/supp(Ĉ) ≥ (1 − α) then
39:     return ;
40:   end if
41: end for
42: print Ĉ;
43: END PROCEDURE
```

**Preset pruning** After calling DCI_MARGIN recursively we only need to consider items from the current $D$ for the margin test in Line 22. Since $\hat{M}$ is empty we already know that none of the items in $P_i$ generate violating supersets. The items in $P \setminus P_i$ are already included in $\hat{C}$, the current closed itemset under study. This leaves any items that were removed from $P$ in Line 4 but not added to $D$ in Line 24. We will now explain why these items can be omitted from the margin test as well. We are only concerned with the case where none of the items in $D$ has already violated the margin condition, i.e., $\forall j \in D\ supp(C \cup i \cup j) < (1-\alpha)supp(C \cup i)$. We can further ignore items that produce infrequent generators and are filtered by the first condition in Line 6 for this computed closure. These would also be ignored in the margin

test in Line 38. This leaves items that produce a frequent generator but have been excluded by the duplicate check. Those can be omitted since those generators are covered by items from $D$.

**Support order pruning** Recall that any total order $\prec$ of the items can be used. DCI_CLOSED works with any fixed order of the items, sorting the items by decreasing support was used in [46]. When used with DCI_MARGIN it increases the probability that the intersection of transaction lists for the support calculation of $\hat{C} \cup j$ in Line 38 results in a violation of the margin condition avoiding further checks. This is especially helpful in dense datasets. Note that this optimization will improve the post-processing algorithm as well.

We refer to [46] for the proof that all returned itemsets are closed. We only need to show that the additional pruning steps do not remove any margin-closed itemsets from the results.

To prove the correctness and completeness of DCI_MARGIN, we need to check whether the delayed pruning decisions for items in $P_i$ for the subsequent call of DCI_MARGIN are correct. Since $P_i \cup \hat{C} \cup D = \mathcal{I}$ is not guaranteed, we also need to show that all items pruned by DCI_CLOSED will not affect the margin decision for subsequent calls. We omitted the proof that there is no need to test infrequent generators for the sake of brevity.

**Lemma 3.1** *Given a closed itemset $C \in \texttt{CFI}$ and items $i, j \in \mathcal{I}$ with $i, j \notin C$, if $g(C \cup i) \subseteq g(j) \Rightarrow \frac{supp(C \cup i)}{supp(C)} \leq \frac{supp(C \cup j)}{supp(C)}$.*

*Proof* $g(C \cup i) \subseteq g(j) \Rightarrow g(C \cup i) = g(C \cup i \cup j) \Rightarrow g(C \cup i) \subseteq g(C \cup j)$ since $g(C \cup i \cup j) \subseteq g(C \cup j) \Rightarrow supp(C \cup i) \leq supp(C \cup j) \Rightarrow \frac{supp(C \cup i)}{supp(C)} \leq \frac{supp(C \cup j)}{supp(C)}$.

The Lemma states, that given a generator $C \cup i$ and an additional item $j$ with $g(C \cup i) \subseteq g(j)$ , we can assume that either the generator $C \cup j$ violates the margin condition or if not neither will $C \cup i$. This allows us to skip the check in line 8 for generators which return true for the duplicate check as long as we use the generator $C \cup j$ for testing the margin-closedness of $C$. Note that this only covers items in $P_i$.

**Lemma 3.2** *Given an infrequent generator $C \cup i$ with $supp(C \cup i) < \theta$ , where $C$ is a closed itemset and $i \in \mathcal{I}$ with $i \notin C$ and another closed itemset $C'$ with $C \subset C'$ which is frequent, i.e., $supp(C') \geq \theta$, then $supp(C' \cup i) < \theta$.*

*Proof* $supp(C \cup i) < \theta \Rightarrow supp(C \cup C' \cup i) < \theta \Rightarrow supp(C' \cup i) < \theta$ since $C \cup C' = C'$.

This allows us to omit all generators $C \cup i$ for TESTMARGIN which were derived from a closed itemset $C$ if $C \cup i$ is not frequent. Therefore we do not need to check the margin condition and can ignore item $i$ for all closed itemsets that are generated by adding items to $C$. In particular, we can avoid adding item $i$ to $D$ for subsequent tests in Line 24 if the generator $C \cup i$ is not frequent (Line 6).

**Lemma 3.3** *Given closed itemsets $C, C' \in \texttt{CFI}$ with $C \subset C'$ and items $i, h \in \mathcal{I}$ with $i, h \notin C'$ then: $g(C \cup i) \subseteq g(h) \Rightarrow \frac{supp(C' \cup i)}{supp(C')} \leq \frac{supp(C' \cup h)}{supp(C')}$*

*Proof* $g(C \cup i) \subseteq g(h) \Rightarrow g(C' \cup i) \subseteq g(h)$ since $g(C' \cup i) \subseteq g(C \cup i) \Rightarrow$[2] $\frac{supp(C' \cup i)}{supp(C')} \leq \frac{supp(C' \cup h)}{supp(C')}$

[2] Using Lemma 3.1.

The Lemma states that pruning item $i$ will not change the result for subsequent margin-checks, since the missing margin-calculation is still covered by item $h$. This allows us to avoid adding $i$ to $D$ in line 24 if the IsDuplicate check returns true and therefore using the pruning technique from DCI_Closed will not lead to incorrect results.

Altogether, Lemma 3.1 allows us to check only those items in $P_i$ that pass the duplicate check if we check all items in $D$. Exploiting Lemma 3.2, we need to check only those items for a closed itemset which did not produce an infrequent generator for a closed subset. Finally, Lemma 3.3 states that the pruning technique used in DCI_Closed will not exclude items necessary for subsequent calculations.

For the proof of correctness and completeness of DCI_Margin we assume that exactly the set of all closed itemsets is presented to the modifications after line 18 due to completeness and correctness of DCI_Closed. We will therefore restrict the proof to itemsets in CFI.

**Corollary 3.4** DCI_Margin *is correct: only margin-closed itemsets are reported.*

*Proof* (by contradiction)

Let be $C \in \texttt{CFI} \setminus \texttt{CFI}^{\alpha}$. Suppose there exists an item $i$, s.t. $C \cup i \in \texttt{FI}$ and $\frac{supp(C \cup i)}{supp(C)} \geq 1 - \alpha$ and $C$ is reported as margin-closed. This leaves two possibilities:

(a) If $i \in P_i \cup D$, we can conclude that $i \in P_i$, since for all $j \in D$ the margin condition is tested in Line 38. Furthermore there must be an item $h \in D$, s.t. $g(C \cup i) \subseteq g(h)$, otherwise $C$ would be checked in Line 8 and $\hat{M}$ would not be empty. This is a contradiction to Lemma 3.1, since $C$ would be checked in TestMargin for item $h$ and therefore not be reported.

(b) $i \in \mathcal{I} \setminus (P_i \cup D)$ Since the algorithm is initiated with the bottom closure as closed itemset and the remaining items as $P$, $i$ was pruned before by either (i) producing an infrequent generator or (ii) the duplicate check returned true for the produced generator.

   (i) Since the generator for which item $i$ was pruned was infrequent, so is $C \cup i$ (according to Lemma 3.2). This is a contradiction to our assumptions.

   (ii) $i$ was not included in $D$ for a closed itemset $C' \subset C$ due to the duplicate detection, which means there exists an item $h \in D$, s.t. $g(C' \cup i) \subseteq g(h)$. Then the closed but not margin-closed itemset would not be reported due to the check in TestMargin in Line 38, since $C \cup i$ will violate the margin condition according to Lemma 3.3.

**Corollary 3.5** DCI_Margin *is complete: all margin-closed itemsets are reported.*

*Proof* Let be $C \in \texttt{CFI}^{\alpha}$. Then we know that $\forall i \in \mathcal{I} \setminus C$ with $C \cup i \in \texttt{FI}$: $\frac{supp(C \cup i)}{supp(C)} < 1 - \alpha$. Since $P_i \cup D \subset \mathcal{I}$, we can deduce that (a) $\forall j \in P_i$ with $C \cup j \in \texttt{FI}$ the generator $C \cup j$ is tested and $\hat{M}$ will be empty in line 21 and (b) $\forall j \in D$ with $C \cup j \in \texttt{FI}$ the test in line 38 will lead to write out $C$. Given that all closed itemsets are traversed we can conclude that all margin-closed itemsets are reported.

### 3.1 Example

For our example, we choose a minimal support $\theta = 0.1$ and $\alpha = 0.1$ as the margin value. To avoid confusion, we use lower case letters to represent the items.

Starting with the example of Fig. 1a and following a lexicographic order, the algorithm is initiated with the empty set as bottom closure. The first item to add is $a$. It is removed from the set of items $P$, which will be processed in subsequent branches of the recursion.

Since it is frequent and no item is in the list $D$, it passes the duplicate check in line 6. In line 7 the margin condition is not violated regarding the previously found closed itemset $\emptyset$. In the loop in line 12 only $c$ and $d$ can be added to the working set $\hat{C}$ since an addition of $b$ would decrease the support. The algorithm has reached the closed itemset $acd$ invoking a recursive function call of DCI_MARGIN. Within this function call only $b$ could be added, but since $abcd$ is infrequent no further processing is necessary after line 6. Since $D$ is empty and no frequent closed superset in this branch exists, we can print out $acd$ as a margin-closed itemset by calling TESTMARGIN. Returning to the subsequent branch of the recursion, $a$ is added to the duplicate list $D$.

Starting again with the empty set, $b$ is added to the working set $C_i$. Since it is frequent and not a subset of $a$ (line 6), $d$ can be added to climb to the closure $bd$. In line 7, the margin condition for the bottom closure $\emptyset$ regarding $b$ is checked. After another call of DCI_MARGIN the newly created working set $C_i = bcd$ does not pass the frequency check. $b$ is added to the duplicate list $D$ in line 24 when returning to the next branch.

Again, calling DCI_MARGIN with the empty set, $c$ is processed and found to be a closed itemset. Since the relative difference between $supp(\emptyset)$ and $supp(c)$ violates the margin condition, $c$ is added to $M$ indicating that the bottom closure is not margin-closed. The algorithm continues with processing the only item left $d$. Since $g(cd)$ is a subset of the first element in $D$, namely $a$, $cd$ is correctly identified as an itemset of a previously visited equivalence class and therefore not added to $D$ before pursuing the last branch. For $c$ the TESTMARGIN procedure is called and it is found to be margin-closed, because none of the items in $D = \{a, b\}$ would result in violating the margin condition.

Finally, $d$ is added in the last branch. Since no other item is left to add in line 12, the TESTMARGIN procedure is called to test the margin condition for the closed itemset $d$. Adding the first item $a$ in $D$ to the closed itemset $d$ passes the frequency check but violates the margin condition in line 38. $d$ is correctly identified as not margin-closed.

The algorithm correctly determines $acd$, $bd$ and $c$ as margin-closed. $abcd$ is infrequent and neither $\emptyset$ or $d$ are margin-closed due to $c$ and $cd$, respectively.

## 4 Experiments

We performed experiments to analyze the class of margin-closed itemsets in detail and to compare the proposed DCI_MARGIN algorithm with a naive extension of DCI_CLOSED. We do not compare experimentally with our own previous work [51] that used a variation of CHARM because this algorithm needs to keep all closed itemsets in memory for the subsumption check. This is a severe disadvantage in particular for low minimum support values. We performed a comparison with FP-Growth variant [22] to demonstrate that the greedy heuristics lead to incorrect results that can vary greatly from the exact result generated by our algorithm.

The implementation was done in Java using bitmap data structures and the correctness was checked using brute force algorithms on small datasets. The experiments were run on a 64-bit dual core Intel Xeon with 2.66 GHz and 8 GB of main memory.

### 4.1 Data sets

We used datasets from three repositories. The FIMI[31] datasets listed in Table 2 include large transaction datasets derived from traffic data, census data, weblogs [38] and retail data. The last two datasets are synthetically generated to simulate market basket data. The datasets

**Table 2** FIMI datasets

| Name | Items | Transactions |
|---|---|---|
| Accidents | 468 | 340,183 |
| BMS-WebView-1 | 59,602 | 149,639 |
| BMS-WebView-2 | 77,512 | 358,278 |
| BMS-POS | 515,597 | 3,367,020 |
| Chess | 75 | 3,196 |
| Connect | 129 | 67,557 |
| Kosarak | 41,270 | 990,002 |
| Mushroom | 119 | 8,124 |
| Pumsb | 2,113 | 49,046 |
| Pumsb∗ | 2,088 | 49,046 |
| Retail | 16,470 | 88,162 |
| T10I4D100K | 870 | 100,000 |
| T40I10D100K | 942 | 100,000 |

from the UCI Machine Learning Repository [4] listed in Table 3 represent classification problems from a wide variety of domains. We used the itemset representations of the datasets taken from the LUCS repository [23]. The text datasets listed in Table 4 are shipped with the Cluto clustering toolkit [83] and were converted to itemsets using a binary representation of words in documents discarding the term frequencies.

### 4.2 Numerosity reduction

A parameter study was performed on all datasets to investigate the redundancy reduction under various minimum support values and minimum margin between 0.005 and 1.0 depending on the dataset. Obviously redundancy can only be removed if there is any. In the interest of space we only show typical examples of datasets where our approach is beneficial and examples for dataset where only few redundancy exists among the closed itemsets.

Figure 2 shows the number of reported itemsets on a log scale vs. the minimum support threshold for some FIMI datasets. The solid line corresponds to all closed itemsets found by DCI_CLOSED (or equivalently $\alpha = 0$). The dashed lines represent DCI_MARGIN with different minimum margin thresholds. The lowest dashed line represents all maximal itemsets ($\alpha = 1.0$).

For all datasets the margin condition can reduce the number of reported itemsets significantly without necessarily reporting only maximal itemsets. In many cases we observe a smooth transition of the curves showing a decreasing number of patterns with increasing $\alpha$.

For Accidents and Pumsb very small margins of 0.01 and 0.005, respectively, already largely reduce the number of itemsets indicating a large number of closed itemsets with very similar frequencies in these datasets. Since these datasets have relatively high minimum support levels, the number of margin-closed itemsets approaches the number of maximal itemsets for values of $\alpha = 0.05$ or larger. For Accidents margins of $0.01 - 0.05$ seem adequate because smaller values do not decrease the number of reported patterns and larger values are equivalent to the maximal itemsets. For Pumsb smaller thresholds up to 0.01 represent a good compromise.

**Table 3** UCI datasets

| Name | Items | Transactions |
|---|---|---|
| Adult | 97 | 48,842 |
| Anneal | 73 | 898 |
| Auto | 137 | 205 |
| Breast | 20 | 699 |
| ChessKRvK | 58 | 28,056 |
| Congres | 34 | 435 |
| Connect4 | 129 | 67,557 |
| CylBands | 124 | 540 |
| Dematology | 49 | 366 |
| *E. coli* | 34 | 1,389 |
| Flare | 39 | 48,842 |
| Glass | 48 | 214 |
| Heart | 52 | 303 |
| Hepatitis | 56 | 155 |
| HorseColic | 85 | 368 |
| Ionosphere | 157 | 351 |
| Iris | 19 | 150 |
| Led7 | 24 | 3,200 |
| LetRecog | 106 | 20,000 |
| Mushroom | 90 | 8,124 |
| Nursery | 32 | 12,960 |
| PageBlocks | 46 | 5,473 |
| PenDigits | 89 | 10,992 |
| Pima | 38 | 768 |
| Soybean-large | 118 | 683 |
| TicTacToe | 29 | 958 |
| Waveform | 101 | 5,000 |
| Wine | 68 | 178 |
| Zoo | 42 | 101 |

For Mushroom, BMS-WebView-2, and T10I4D100K the margins of 0.01 and 0.05 lead to a significant reduction. Larger margins continue to reduce the number of patterns in a relatively smooth transition.

T40I10D100k, Retail, and BMS-WebView-1 show a different behavior. For the Retail data margins up to 0.25 reduce the number of reported itemsets but by far not as clearly as for the other datasets. There does not seem to be a lot of redundancy which could be caused by the large number of distinct items. Margins between 0.1 and 0.5 would be useful to reduce the size of the result. For the T40I10D100k data the number of margin-closed itemsets is very close to the number of maximal itemsets for $\alpha \geq 0.05$ and small minimum supports. For larger minimum support thresholds the number of closed and maximal itemsets are much more similar with margin-closed itemsets providing a better transition.

Figure 3 shows the number of reported itemsets for some UCI datasets. Adult is an example for cases where increasing the margin threshold leads to a gradual reduction of patterns.

**Table 4** Text datasets

| Name | Items | Transactions |
|---|---|---|
| Cacmcisi | 41,681 | 4,663 |
| Classic | 41,681 | 7,094 |
| Cranmed | 41,681 | 2,431 |
| Fbis | 2,000 | 2,462 |
| Hitech | 126,373 | 2,301 |
| K1a | 21,839 | 2,340 |
| K1b | 21,839 | 2,340 |
| La1 | 31,472 | 3,204 |
| La12 | 31,472 | 6,279 |
| La2 | 31,472 | 3,075 |
| Mm | 126,373 | 2,521 |
| New3 | 83,487 | 9,557 |
| Ohscal | 11,465 | 11,161 |
| Re0 | 2,886 | 1,503 |
| Re1 | 3,758 | 1,656 |
| Reviews | 7,454 | 4,069 |
| Sports | 8,261 | 8,580 |
| Wap | 8,460 | 1,559 |

Note that the interplay of support and margin can lead to non-monotone curves, but in general the larger the minimum support and the larger the minimum margin, the less patterns are reported. Hepatitis is an example where the redundancy removal is very effective for low minimum supports but vanishes for larger values. Again, this is due to the large absolute values of the minimum support of 60–70% that does not leave a lot of room for any margin of support. For ChessKRvK using a margin constraint does not change the result significantly because no redundancy could be detected.

Figure 4 shows the number of reported itemsets for some text datasets. The first two datasets are examples for document collections where the margin constraint can successfully remove redundancy in the word combination patterns. The last dataset is an example for very concise sets of patterns where even large margins do not reduce the number of patterns.

### 4.3 Computational complexity

We compare DCI_MARGIN with a naive version of mining margin-closed patterns that adds a post-processing check to DCI_CLOSED to evaluate the efficiency of our pruning steps. The post-processing extends each closed pattern by all items not in the pattern and checks the margin condition. This post-processing takes advantage of the support order pruning, but not delay or preset pruning.

The algorithms can be implemented efficiently using bitmaps to store the transactions for each item. The high level operations on the bitmaps include logical AND (when adding an item to an itemset), subset (checking if an item can be added to an itemset without decreasing the support) and cardinality (determining the support of an itemset). In order to be independent of influences of operating system, programming language, and just-in-time compilers

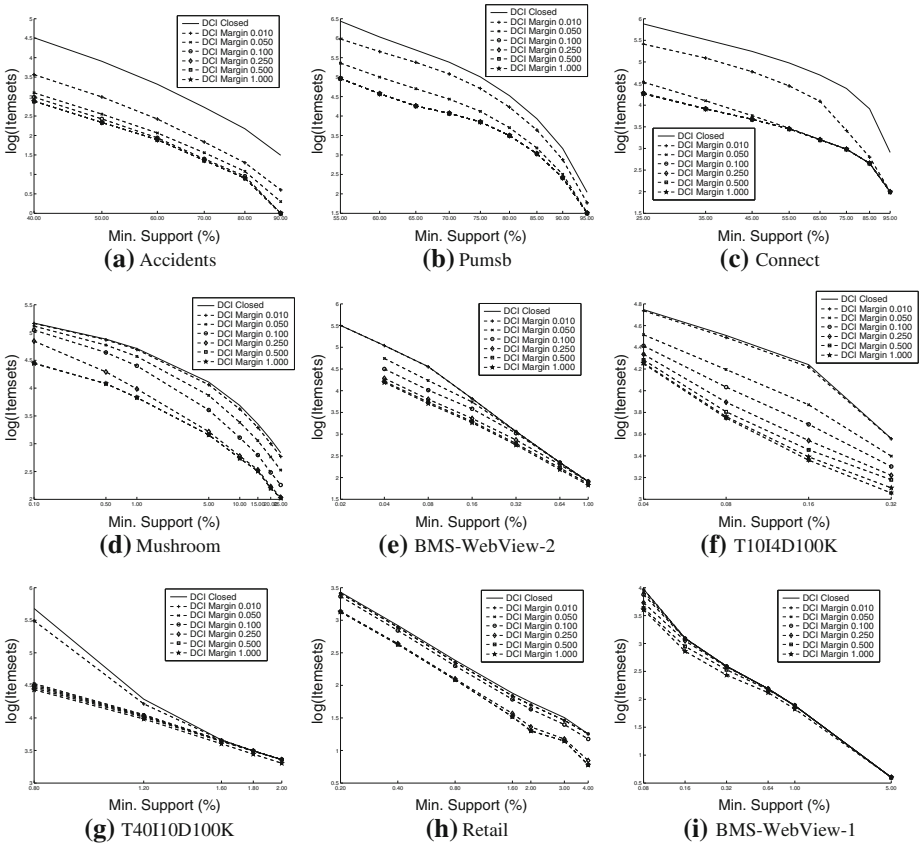



**Fig. 2** Number of (margin-)closed itemsets for different minimum support and minimum margin in FIMI datasets

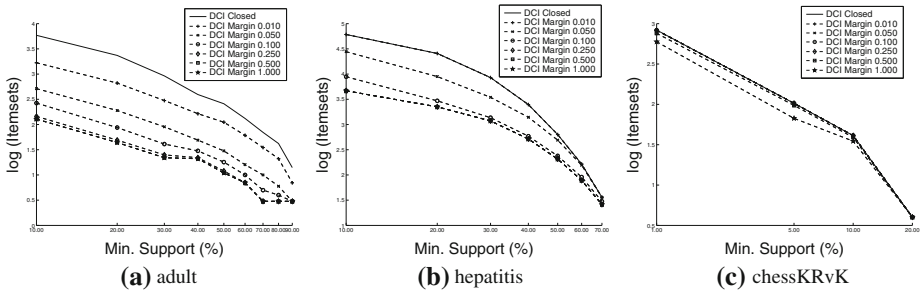
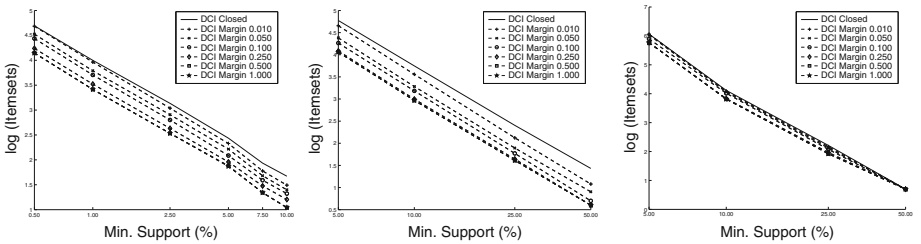


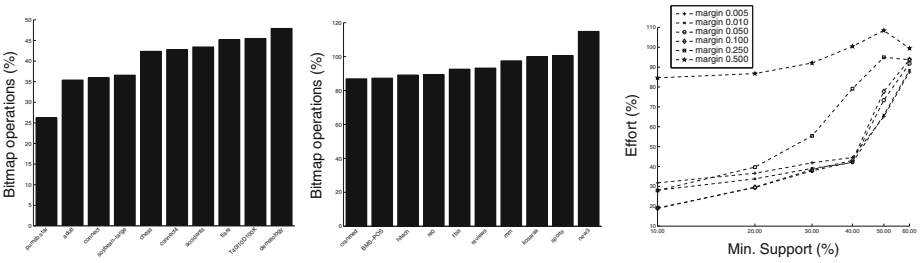

**Fig. 3** Number of (margin-)closed itemsets for different minimum support and minimum margin in UCI datasets

we evaluated the algorithms using counters for the bitmap operations that dominate the computational effort. All bitmaps are accompanied by a pointer to the position of the last set bit. All three high level operations use these pointer to return early. The subset operation is further aborted early as soon as a violating bit is discovered. We counted all low-level 64-bit word operations corresponding to these high level operations.




**(a)** classic **(b)** hitech **(c)** new3

**Fig. 4** Number of (margin-)closed itemsets for different minimum support and minimum margin in text datasets



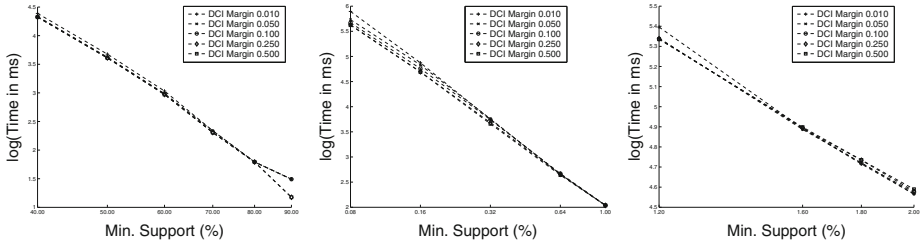

**(a)** Best **(b)** Worst **(c)** Pumsb_star

**Fig. 5** Relative number of bit operations of DCI_MARGIN compared to naive post-processing

In Fig. 5, we show the relative effort required by DCI_MARGIN compared to DCI_CLOSED with post-processing. Values below 100% thus indicate that the pruning methods increased the efficiency for the particular choice of parameters and dataset. For each dataset we summed up the effort over all minimum support levels used in Sect. 4.2 and minimum margins 0.01 through 0.25 as would be typically used in practice. The ten best performances of DCI_MARGIN in Fig. 5a requires only 25–50% of the bit vector operations of DCI_CLOSED with post-processing. The ten worst performances of DCI_MARGIN show the same or slightly worse performance than the post-processing. More bit operations than post-processing can be observed if many delay or preset pruning steps are performed unsuccessfully without removing closed itemsets. For the best dataset the effort is shown for each minimum support and minimum margin level in Fig. 5c.

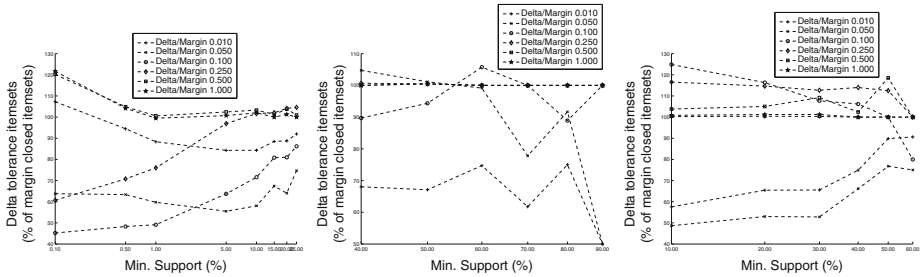
The results indicate a clear benefit of the proposed pruning steps. The observed gains in effort highly depend on the dataset. In the best cases a significant amount of computation is saved and in the worst cases the performance is similar to post-processing.

The space and time complexity of DCI_MARGIN are inherited from DCI_CLOSED [46]. The space complexity is bounded by the size of the dataset and independent of the number of patterns found. No explicit computational analysis was given in [46] but the runtimes compared favorably with other efficient algorithms. Furthermore, our algorithm inherits important scalability properties from DCI_CLOSED the possibility to parallelize the search space traversal [48] and mine the data out of core [47]. In Fig. 6 we show the actual run times for the largest datasets and several minimum support and margin thresholds. For very low minimum supports on the large dataset BMS-WebView 2 the mining took about 15 min. As for all itemset algorithms the runtime increases exponentially with lower minimum supports. Different values of the margin have relatively small influence on the runtime.




**(a)** Accidents **(b)** BMS WebView 2 **(c)** T40I10D100K

**Fig. 6** Runtime of DCI_MARGIN for selected large datasets. For the most complex problem under study the mining took up to about 15 min




**(a)** Mushroom **(b)** Accidents **(c)** Pumsb∗

**Fig. 7** Number of reported $\delta$-tolerance itemsets as percentage of the correct number of $\delta$-tolerance closed (or equivalently margin-closed) itemsets

### 4.4 $\delta$-tolerance itemsets

We obtained the binary version of the algorithm in [22] and compared the output with the complete set of margin-closed itemsets given the same minimum support and threshold (our margin and their $\delta$). The results for three datasets evaluated are shown in Fig. 7. In many cases the algorithm is observed to report a significant number of non-margin-closed itemsets (greater than 100% on the $x$ axis) or to report less than the true number of margin-closed itemsets (less than 100%) ranging from under 50% to more than 120%. There is no clear trend in the behavior of the algorithm with respect to minimum support or margin. While for large minimum support thresholds the absolute difference in reported itemsets is small, for more complex settings the difference can be huge: For the lowest minimum support and margin on Pumsb∗ only 133490 of all 231882 margin-closed itemsets are reported. This demonstrates the greedy heuristic can lead to results that vary significantly from the correct result reported by our algorithm.

## 5 Applications

In this section, we demonstrate the usefulness of margin-closed itemsets in two applications. In exploratory analysis of temporal patterns the removal of redundancy generates better interpretable results. For compression-based data mining tasks better understandable codebooks with comparable performance are generated.

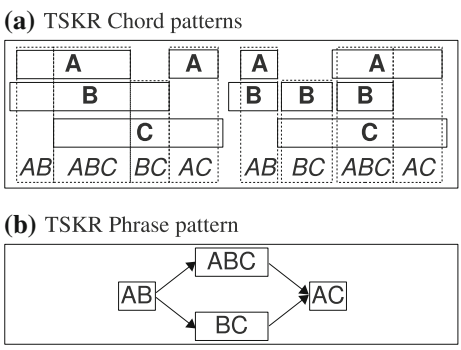



**Fig. 8** Simultaneous occurring intervals (Tones) form Chords. Phrases describe a partial order of Chords

### 5.1 Temporal data mining

In [52], temporal patterns based on the Time Series Knowledge Representation (TSKR) are mined from symbolic interval time series. The data model for this method consists of possibly overlapping time intervals with a label. Hierarchical patterns are defined as groups of intervals simultaneously active on a subinterval (Chords) on the first level, and a partial order of such groups (Phrases) on the second level. Figure 8a shows a series of observed intervals with labels $A$, $B$, and $C$ and intervals of Chord patterns such as $ABC$ beneath. Figure 8b shows a Phrase that represents a partial order of Chords matching both blocks of intervals in Fig. 8a. The mining of TSKR patterns can be formulated as a combination of itemset and sequential pattern mining in three phases: (1) Closed itemsets are mined interpreting the interval labels as items and time units as transactions. (2) Sequential patterns are mined interpreting Chords as items and intervals between any Chord start and end point as transactions in a sequence. (3) Closed itemsets are mined interpreting sequential patterns as items and sequences as transactions. Finally, each closed group of sequential patterns is converted to a partial order graph [28].

The support of a Chord is the total number of time units where the intervals are observed simultaneously. Small differences in support are typically meaningless, such as the short leading or trailing parts of observed intervals not described by a Chords in Fig. 8a. Margin-closed itemsets can be used to mine a smaller set of Chords with less redundancy. Chords with almost the same duration than more specific Chords are pruned. This leads to better interpretable results and reduces the complexity of the sequential pattern mining algorithm. Groups of simultaneous sequential pattern form a closed partial order [28]. Again, margin-closedness leads to a reduction of the reported partial orders pruning less specific patterns that are observed in only few additional sequences.

In an application to sports medicine using a minimum margin of 0.1 reduced number of Chords from 60 to 18 and the number of Phrases from 20 to 15. The absolute numbers are small but they significantly eased the burden of the manual analysis by an expert. Having to analyze many very similar patterns can easily result in frustration of the analysts. The expert selected the Phrase in Fig. 9 as the most interesting pattern that describes the muscle activation during inline speed skating. The Chord *Push Gluteus* is expanded to show the corresponding items and for the item *Gluteus is very high* the represented value range in the original numerical time series is shown.

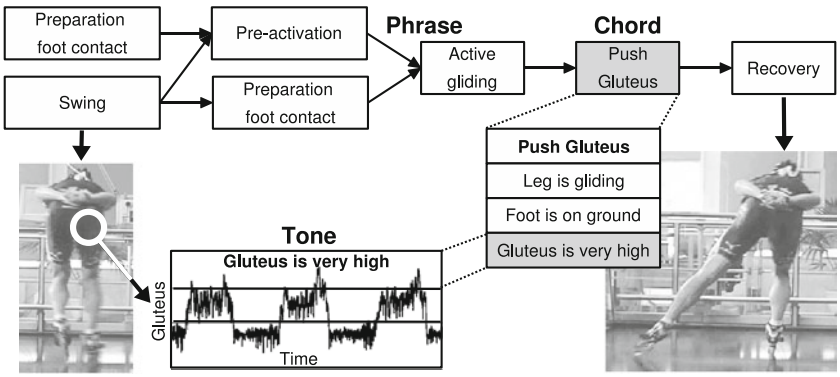



**Fig. 9** Detailed Phrase of skating data with additional information on muscle activation

## 5.2 Mining by compression

Performing data mining tasks using compression as an approximation to Kolmogorov complexity has recently gained popularity, see [25,37] and references therein. For itemsets the Krimp algorithm has been proposed to find a codebook of itemsets that compresses a transaction database well [63]. The algorithm has subsequently been used to support classification [71], change detection [70], and missing value replacement [72].

Krimp starts with a trivial code book of single items that is greedily improved considering longer itemsets from a set of candidate patterns. The authors recommend to use closed itemsets rather than frequent itemsets to avoid redundancy. We performed a set of experiments to evaluate the use of margin-closed itemsets as candidates.

We ran the Krimp algorithm with closed and margin-closed itemsets as candidates and compared the sizes of the compressed database. We used the smallest minimum support values and the same minimum margin values as in Sect. 4.2. Only the UCI datasets were used because compression is particularly relevant for classification and the available version of the Krimp program had problems processing the high dimensional sparse text datasets.

Figure 10 plots the ratio of candidates vs. the ratio in achieved compression. The ratios are calculated comparing the margin-closed itemsets with closed itemsets as the baseline. For example, for the TicTacToe data using only 35% of the closed itemsets as candidates results in a compressed database that is only 2.5% bigger. For the sake of presentation, we show only the best result over all evaluated margin values for most datasets. The best result for each dataset was determined using the minimum sum of the candidate ratio and compression ratio, i.e., using the Manhattan distance from (0, 100).

For many datasets, a significant reduction in the number of candidates does not hurt the compression. This directly translates into much improved runtime of the Krimp algorithms that uses the candidate itemsets to build a codebook that compresses the data best. A larger number of datasets has less than 10% increase in compression with fewer than 30% of the closed itemsets. In some cases we can even improve the compression ratio. Most notably, for the Zoo dataset we achieve a slightly better compression using less than 5% of the closed itemsets. The dotted lines passing through the best results for Nursery, Iris, Led7, and Adult show the candidate and compression ratios for multiple margin values applied to these datasets. This shows how different margin values provide a trade off between candidate size and compression.

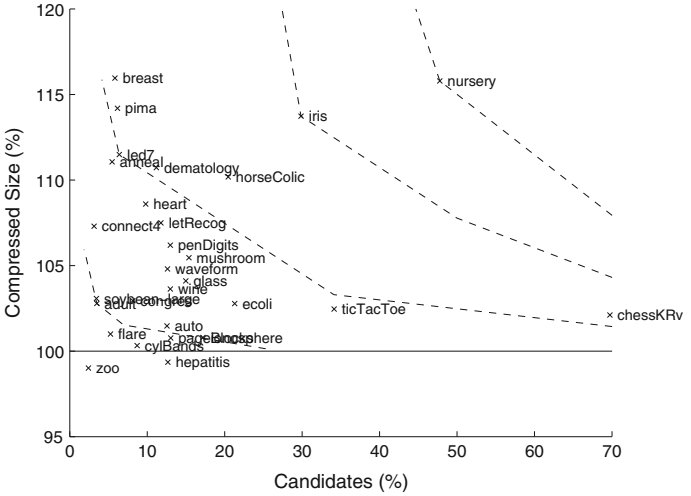



**Fig. 10** Relative number of candidates vs. relative size of compressed database when using margin-closed itemsets instead of closed itemsets as candidates for Krimp

The compression results show that a lot of redundancy can be removed without compromising the quality of the codebook. The reduced number of candidates speeds up the codebook generation of Krimp and makes the codebook more interpretable. For example, to understand why a certain instance has been labeled by the Krimp-based classifier in a particular way one can extract the codebook vectors that had the largest influence on the decision. We expect that the performance of subsequent data mining tasks will not suffer significantly because comparable compression ratio are obtained.

## 6 Discussion

Our experimental results show that with comparable run time our algorithm can mine the more compact set of all margin-closed itemsets instead of reporting all closed itemsets. The pruning is performed on-the-fly utilizing the data structures of DCI_CLOSED and saving IO costs otherwise required to report all closed itemsets. The best value of $\alpha$ is application dependent: which difference in support between similar patterns can be considered insignificant enough to report only the longer pattern? In most cases, we assume that $\alpha$ would be small but not very small (between 0.01 and 0.2). For very small $\alpha$, one should expect increased run time because less pruning can be performed. If the margin is chosen to be bigger than $1 - \theta$ only maximal itemsets are reported. In some applications, it might be more natural to specify an absolute support margin. All our results hold and the algorithm can be used in the same way.

In principle, the test for margin-closedness can also be integrated in the FP-Tree [35] algorithms or the version of DCI_CLOSED for sparse datasets. We chose the DCI_CLOSED algorithm for dense datasets as the basis of our work for several reasons. The vertical data format used by DCI algorithms can be exploited with the SIMD architecture of modern processors and even GPUs. In addition, the found patterns do not need to stay in memory and the partition of the search space enables parallelization [48]. The vertical representation has

further advantages for itemset problems that represent temporal data [52] which is typically dense. Checking additional constraints on the duration of temporal patterns can be easily done using bit vectors but would require tracking the transaction times in projected databases [35].

A breadth-first approach might be more suitable for sparse datasets [77]. Since the number of closed itemsets is commonly much smaller in this case the margin condition can be checked after the closed itemset computation.

The concept of a minimum margin could also be used to generalize the definition of minimal generators [41], the minimal elements of an equivalence class induced by the closure operator. A minimal generator is an itemset where no item can be removed without increasing the support. A margin minimal generator would be an itemset where no itemset can be removed without increasing the support significantly (given a threshold parameter).

Our aim was to avoid redundancy of reported patterns to support exploratory analysis and favor longer patterns with more explanatory power. The concept of margin-closedness is in no way limited to itemsets, it can also be applied to sequential patterns [3], partial orders [60,52] and graphs [40].

## 7 Related work

The two closest publications to our approach are $\delta$-tolerance itemsets [22] and relaxed frequent closed itemsets [64].

The $\delta$-tolerance closed itemsets of [22] are equivalent in definition to margin-closed itemsets and have been proposed independently. The motivation in [22] was to provide a condensed itemset representation that provides an approximate frequency estimation for itemsets. This is achieved with approximation formulas that use the support and the support differences (margins) of the itemsets stored in an FP-tree [35]. The mining algorithm uses several heuristics that try to avoid but do not guarantee false dismissals. The reported itemsets are thus possibly a subset of all margin-closed itemsets. As demonstrated, this does not seem to hurt the frequency estimation and enables fast performance. In contrast, we can guarantee completeness, which is important for exploratory analysis. While not designed for frequency estimation, the same techniques as proposed in [22] are applicable to our approach.

The relaxed frequent closed itemsets of [64] require the user to define a uniform partition of the support range. Subsets whose supersets are in the same support interval are pruned removing redundancy. The motivation is to reduce the number of patterns in memory when mining data streams. The effectiveness and efficiency was demonstrated using synthetic data. The a priori definition of several support thresholds might still generate redundant patterns if the supports of the subset and superset are just below and above one of the support thresholds, respectively. In contrast our pruning is data driven and removes any redundancy according to the single threshold $\alpha$.

In comparison with the two approaches outlined above, we performed much more extensive experiments with a total of 60 datasets from many different domains whereas [22] and [64] used only three and one FIMI datasets, respectively.

In our previous work, we have presented a modified CHARM [82] algorithm to mine margin-closed itemsets [51]. This approach suffers from scalability problems because it requires all closed (even the non-margin-closed) itemsets to be kept in memory for the subsumption check.

We proceed to categorize less directly related approaches below by the purpose they have been designed for to highlight the differences to our approach.

Condensed itemset representations [18] have been developed to derive the support of all itemsets from a compact summary exactly [13,39,14,15,53,16,44] or approximately [57,11,22]. Querying the support of an itemset from a data structure is a key step in generating association rules [36]. The basic idea of non-derivable itemsets [16] and related approaches is to derive the support of a query itemset from the support of subsets stored in the condensed representation [14,15]. If this is possible exactly or within error bounds, the larger set does not need to be stored in the summary. [61] prunes all supersets with approximately the same support as a smaller itemset. Note that these approaches favor short itemsets and prune longer itemsets, whereas we prune the shorter subsets with support similar to a longer supersets. This favors more detailed patterns that are generally more interesting in exploratory analysis.

Condensed representations are a special case of more general constraints on the reported itemsets that are commonly categorized into several classes [65,54]: monotone, anti-monotone, succinct, convertible and tough. The first three were integrated in early constraint-based itemset mining algorithms. Convertible constraints were later integrated for depth first algorithms in [58] and for level-wise algorithms in [7], see [9] for more details. [8] describes issues with combining closed itemsets (and other condensed representations) with additional constraints. Our margin constraint does not cut closure equivalence classes but simply merges them avoiding potential problems. Recently, [24] presented an elegant way of mining constrained itemsets, including margin-closed itemsets, with constraint programming.

A related line of work is motivated by the fact, that transaction data is often noisy. The strict definition of support, requiring all items of an itemset to be present in a transaction, is relaxed [59,79,62,1,78,45,20,69,17,21]. A recent comparison analyzed the efficiency and effectiveness of approximate itemset mining [33]. These approaches can reveal important structures in noisy data that might otherwise get lost in a huge amount of fragmented patterns. One needs to be aware though that they report approximate support values and possibly list itemsets that are not observed as such in the collection at all [1] or with much smaller support. This might be misleading in exploratory applications. In our application of itemset mining to temporal data mining [52] we filtered out noise in preprocessing steps using the temporal structure of the data and found it beneficial to list exact patterns with exact support. This corresponds to the Gricean maxim of quality [32] that states that only well supported facts and no false descriptions should be reported and has been recommended as a guideline for pattern discovery for data exploration [66]. Finally, we want to note that margin-closed itemsets might be used instead of closed itemsets as seeds to the AC-Close algorithm for approximate itemset mining [20] improving its efficiency that was criticized in [33].

Other approaches try to reduce the number of patterns after they are mined [1,76,50]. By this time a lot of computational resources have been spent on mining and storing the results. Our algorithm integrates the mining with the pruning on-the-fly and never stores or further processes the superfluous patterns.

For post-processing techniques such as [12] or [63] that use closed itemsets as their input and remove redundancy in the pattern set, margin-closed itemsets can be used as an alternative input reducing their runtime without sacrificing performance. This was demonstrated for [63] in Sect. 5.2. In [12] a small subset of patterns is selected that preserves much of the transaction partition collectively induced by presence and absence of a set of patterns. Patterns are selected according to a user defined ordering such as by size or by support.

In [30], the number of reported closed itemsets is reduced to the top-k patterns optimizing the coverage of the database with the transactions and items of the patterns. This is likely to remove redundancy in the output but our constraint is more explicit. It would be interesting

to investigate the top-k least redundant pattern mining problem. In [6] the number of frequent (but not closed) itemsets given a minimum support is estimated using random walks on the itemset lattice.

In addition to the margin constraint, statistical measures for interestingness, significance, and surprise [49,67,27,75,68] could be used to rank or further reduce the number of reported margin-closed itemsets.

In summary, margin-closedness is a stricter constraint than closedness that leads to a lossy, concise, exact itemset representation designed for exploratory and explanatory data mining tasks.

## 8 Summary

Margin-closed itemsets provide a compromise between closed and maximal itemsets designed for exploratory data analysis favoring longer itemsets that provide the users with more specific information and reporting exact information. We have presented DCI_MARGIN, a new efficient algorithm that mines *all* margin-closed itemsets on-the-fly and proved its correctness and completeness. Compared to closed itemset mining the algorithm can largely reduce the number of reported itemsets depending on the redundancy structure of the dataset under study. The algorithm achieves this with small computational overhead and was experimentally shown to have comparable or better speed than DCI_CLOSED. We show the usefulness of the patterns in two applications: exploratory mining for temporal patterns [52] and finding compressing datasets [63] that are useful for classification, change detection, or missing value replacement.

**Acknowledgments** We thank Matthijs van Leeuwen and James Cheng for sharing their software and Philipp Hussels for helping to run it. We acknowledge Blue Martini Software for contributing the KDD Cup 2000 data.

## References

1. Afrati F, Gionis A, Mannila H (2004) Approximating a collection of frequent sets, In: Proceedings of 10th ACM SIGKDD international conference on Knowledge Discovery and Data Mining. ACM Press, pp 12–19
2. Agrawal R, Imielinski T, Swami AN (1993) Mining association rules between sets of items in large databases. In: Proceedings of ACM SIGMOD international conference on Management of Data. ACM Press, pp 207–216.
3. Agrawal R, Srikant R (1995) Mining sequential patterns. In: Yu PS and Chen ASP (eds) Proceedings of 11th international conference on data engineering. pp 3–14
4. Asuncion A, Newman D (2007) UCI machine learning repository. http://www.ics.uci.edu/~mlearn/MLRepository.html
5. Beil F, Ester M, Xu X (2002) Frequent term-based text clustering, In: Proceedings of 8th international conference on knowledge discovery and data mining. pp 436–442
6. Boley M, Grosskreutz H (2009) Approximating the number of frequent sets in dense data. Knowl Inf Syst 21(1):65–89
7. Bonchi F, Lucchese C (2005) Pushing tougher constraints in frequent pattern mining, In: Proceedings of Pacific-Asia conference on knowledge discovery and data Mining. pp 114–124
8. Bonchi F, Lucchese C (2006) On condensed representations of constrained frequent patterns. Knowl Inf Syst 9(2):180–201
9. Bonchi F, Lucchese C (2007) Extending the state-of-the-art of constraint-based pattern discovery. Data Min Knowl Discov 60(2):377–399

10. Boulicaut J-F, Bykowski A (2000) Frequent closures as a concise representation for binary data mining. In: Proceedings of Pacific-Asia conference on knowledge discovery and data mining. pp 62–73
11. Boulicaut J-F, Bykowski A, Rigotti C (2003) Free-sets: a condensed representation of boolean data for the approximation of frequency queries. Data Min Knowl Discov 7(1):5–22
12. Bringmann B, Zimmermann A (2009) One in a million: picking the right patterns. Knowl Inf Syst 18(1):61–81
13. Bykowski A, Rigotti C (2001) A condensed representation to find frequent patterns. In: Proceedings of 20th ACM SIGMOD-SIGACT-SIGART symposium on principles of database systems. ACM Press, pp 267–273
14. Calders T, Goethals B (2002) Mining all non-derivable frequent itemsets. In: Proceedings of 6th European conference on principles of data mining and knowledge discovery. Springer, pp 74–85
15. Calders T, Goethals B (2003) Minimal *k*-free representations of frequent sets, In: Proceedings of 7th European conference on principles and practice of knowledge discovery in databases. Springer, pp 71–82
16. Calders T, Goethals B (2007) Non-derivable itemset mining. Data Min Knowl Discov 14(1):171–206
17. Calders T, Goethals B, Mampaey M (2007) Mining itemsets in the presence of missing values. In: Proceedings of international symposium on applied computing. ACM, pp 404–408
18. Calders T, Rigotti C, Boulicaut J-F (2006) A survey on condensed representations for frequent sets. In: Constraint-based mining and inductive databases. pp 64–80
19. Cheng H, Yan X, Han J, Hsu C (2007) Discriminative frequent pattern analysis for effective classification. In: Proceedings of IEEE international conference on data engineering. pp 716–725
20. Cheng H, Yu PS, Han J (2006) AC-Close: efficiently mining approximate closed itemsets by core pattern recovery. In: Proceedings of IEEE international conference on data mining. IEEE pp 839–844
21. Cheng H, Yu PS, Han J (2008) Approximate frequent itemset mining in the presence of random noise. In: Soft computing for knowledge discovery and data Mining. Springer, pp 363–389
22. Cheng J, Ke Y, Ng W (2006) $\delta$-tolerance closed frequent itemsets. In: Proceedings of 6th IEEE international conference on data mining. IEEE Press, pp 139–148
23. Coenen F (2003) The LUCS-KDD discretised/normalised ARM and CARM data library. http://www.csc.liv.ac.uk/~frans/KDD/Software/LUCS_KDD_DN/
24. De Raedt L, Guns T, Nijssen S (2008) Constraint programming for itemset mining. In: Proceedings of 14th ACM SIGKDD international conference on knowledge discovery and data mining. ACM, pp 204–212
25. Faloutsos C, Megalooikonomou V (2007) On data mining, compression, and kolmogorov complexity. Data Min Knowl Discov 15(1):3–20
26. Fung B, Wang K, Ester M (2003) Hierarchical document clustering using frequent itemsets, In: Proceedings of SIAM international conference on data mining
27. Gallo A, De Bie T, Cristianini N (2007) Mini: Mining informative non-redundant itemsets, In: Proceedings of European symposium on principles of data mining and knowledge Discovery. pp 438–445
28. Garriga G (2005) Summarizing sequential data with closed partial orders. In: Proceedings of 5th SIAM international conference on data mining. SIAM, pp 380–391
29. Garriga G, Kralj P, Lavrac N (2006) Closed sets for labeled data. In: Proceedings of European conference on principles and practice of knowledge discovery in databases. pp 163–174
30. Geerts F, Goethals B, Mielikäinen T (2004) Tiling databases. In: Proceedings of discovery science. pp 278–289
31. Goethals B, Zaki M (2003) FIMI '03, frequent itemset mining implementations, In: Proceedings of ICDM 2003 workshop on frequent itemset mining implementations
32. Grice H (1989) Studies in the way of Words. Harvard University Press, Cambridge
33. Gupta R, Fang G, Field B, Steinbach M, Kumar V (2008) Quantitative evaluation of approximate frequent pattern mining algorithms. In: Proceedings of 14th ACM SIGKDD international conference on knowledge discovery and data Mining. ACM, pp 301–309
34. Han J, Pei J (2001) Pattern growth methods for sequential pattern mining: Principles and extensions, In: Workshop on temporal data mining, 7th ACM SIGKDD international conference on knowledge discovery and data mining. ACM Press
35. Han J, Pei J, Yin Y (2000) Mining frequent patterns without candidate generation. In: Proceedings of ACM SIGMOD international conference on management of data. ACM Press, pp 1–12
36. Hipp J, Güntzer U, Nakhaeizadeh G (2000) Algorithms for association rule mining—a general survey and comparison. SIGKDD Explor 2(1): 58–64
37. Keogh E, Lonardi S, Ratanamahatana C, Wei L, Lee S, Handley J (2007) Compression-based data mining of sequential data. Data Min Knowl Discov 14(1):99–129
38. Kohavi R, Brodley C, Frasca B, Mason L, Zheng Z (2000) 'KDD-Cup 2000 organizers' report: Peeling the onion. SIGKDD Explor 2(2): 86–98. http://www.ecn.purdue.edu/KDDCUP

39. Kryszkiewicz M (2001) Concise representation of frequent patterns based on disjunction-free generators, In: Proceedings of 1st IEEE international conference on data mining. IEEE Press, pp 305–312
40. Kuramochi M, Karypis G (2001) Frequent subgraph discovery. In: Proceedings of IEEE international conference on data mining, pp 313–320
41. Li J, Li H, Wong L, Pei J, Dong G (2006) Minimum description length principle: generators are preferable to closed patterns. In: Proceedings of AAAI, pp 409–414
42. Li W, Han J, Pei J (2001) CMAR: Accurate and efficient classification based on multiple class-association rules. In: Proceedings of IEEE international conference on data mining. pp 369–376
43. Liu B, Hsu W, Ma Y (1998) Integrating classification and association rule mining. In: Proceedings of international conference on knowledge discovery and data mining. pp 80–86
44. Liu G, Li J, Wong L (2008) A new concise representation of frequent itemsets using generators and a positive border. Knowl Inf Syst 17(1):35–56
45. Liu J, Paulsen S, Wang W, Nobel A, Prins J (2005) Mining approximate frequent itemsets from noisy data. In: Proceedings of 5th international conference data mining. IEEE, pp 721–724
46. Lucchese C, Orlando S, Perego R (2006a) Fast and memory efficient mining of frequent closed itemsets. IEEE Trans Knowl Data Eng 18(1):21–36
47. Lucchese C, Orlando S, Perego R (2006b) Mining frequent closed itemsets out of core, In: Proceedings of the 6th SIAM international conference on data mining (SDM'06)
48. Lucchese C, Orlando S, Perego R (2007) Parallel mining of frequent closed patterns: harnessing modern computer architectures. In: Proceedings IEEE international conference on data mining
49. Malik H, Kender J (2006) High quality, efficient hierarchical document clustering using closed interesting itemsets. In: Proceedings of IEEE international conference on data mining. pp 991–996
50. Mielikäinen T. (2005) Summarization techniques for pattern collections in data mining, PhD thesis, University of Helsinki, Finland
51. Mörchen F (2006) Algorithms for time series knowledge mining, In: Proceedings 12th ACM SIGKDD international conference on knowledge discovery and data mining. ACM Press, pp 668–673
52. Mörchen F, Ultsch A (2007) Efficient mining of understandable patterns from multivariate interval time series. Data Min Knowl Discov 15(2):181–215
53. Muhonen J, Toivonen H (2006) Closed non-derivable itemsets. In: Proceedings European symposium on principles of data mining and knowledge discovery. pp 601–608
54. Ng R, Lakshmanan LV, Han J, Pang A (1998) Exploratory mining and pruning optimizations of constrained associations rules. In: Proceedings of ACM SIGMOD conference on management of Data. ACM, pp 13–24
55. Nijssen S, Fromont E (2007) Mining optimal decision trees from itemset lattices. In: Proceedings of international conference on knowledge discovery and data mining. ACM, pp 530–539
56. Pasquier N, Bastide Y, Taouil R, Lakhal L (1999) Discovering frequent closed itemsets for association rules. In: Proceedings of 7th international conference on database theory. Springer, pp 398–416
57. Pei J, Dong G, Zou W, Han J (2002) On computing condensed frequent pattern bases. In: Proceedings of 2nd IEEE international conference on data mining. IEEE Press, pp 378–385
58. Pei J, Han J, Lakshmanan LVS (2001) Mining frequent itemsets with convertible constraints. In: Proceedings of IEEE international conference on data Engineering. IEEE, pp 433–442
59. Pei J, Tung AK, Han J (2001) Fault-tolerant frequent pattern mining: problems and challenges. In: Workshop on research issues in data mining and knowledge discovery, 20th ACM SIGMOD-SIGACT-SIGART symposium on principles of database systems
60. Pei J, Wang H, Liu J, Wang K, Wang J, Yu P (2006) Discovering frequent closed partial orders from strings. IEEE Trans Knowl Data Eng 18(11):1467–1481
61. Pudi V, Haritsa J (2003) Generalized closed itemsets for association rule mining. In: Proceedings of 19th international conference on data engineering. IEEE Press pp 714–716
62. Seppänen J, Mannila H (2004) Dense itemsets. In: Proceedings of 10th ACM SIGKDD international conference on knowledge discovery and data mining. ACM Press, pp 683–688
63. Siebes A (2006) Item sets that compress. In: Proceedings of SIAM Conference on data mining. pp 393–404
64. Song G, Yang D, Cui B, Zheng B, Liu Y, Xie K (2007) CLAIM: An efficient method for relaxed frequent closed itemsets mining over stream data. In: Proceedings of 12th international conference on database systems for advanced applications. Springer, pp 664–675
65. Srikant R, Vu Q, Agrawal R (1997) Mining association rules with item constraints, In: Proceedings of international conference on knowledge discovery and data mining. ACM, pp 67–73
66. Sripada SG, Reiter E, Hunter J (2003) Generating English summaries of time series data using the Gricean maxims, In: Proceedings of 9th ACM SIGKDD international conference on knowledge discovery and data mining. ACM Press, pp 187–196

67. Tatti N (2007) Maximum entropy based significance of itemsets. In: Proceedings of IEEE international conference on data mining. pp 312–321
68. Tatti N (2008) Maximum entropy based significance of itemsets. Knowl Inf Syst 17(1):57–77
69. Uno T, Arimura H (2007) An efficient polynomial delay algorithm for pseudo frequent itemset mining. In: Proceedings of 10th international conference discovery science. Springer, pp 219–230
70. Van Leeuwen M, Siebes A (2008) StreamKrimp: Detecting change in data streams. In: Proceedings of European conference on machine learning and principles and practices of knowledge discovery in data. pp 765–774
71. van Leeuwen M, Vreeken J, Siebes A (2006) Compression picks item sets that matter, In: Proceedings of European conference on principles and practice of knowledge discovery in databases. pp 585–592
72. Vreeken J, Siebes A (2008) Filling in the blanks—Krimp minimisation for missing data. In: Proceedings of 8th IEEE international conference on data mining. pp 1067–1072
73. Wang J, Karypis G (2006) On mining instance-centric classification rules. IEEE Trans Knowl Data Eng 18(11):1497–1511
74. Wang K, Xu C, Liu B (1999) Clustering transactions using large items. In: Conference on information and knowledge management. pp 483–490
75. Webb GI (2007) Discovering significant patterns. Mach Learn 68(1):1–33
76. Xin D, Han J, Yan X, Cheng H (2005) Mining compressed frequent-pattern sets. In: Proceedings of 31st international conference on very large data bases. pp 709–720
77. Yahia SB, Hamrouni T, Mephu Nguifo E (2006) Frequent closed itemset based algorithms: a thorough structural and analytical survey. ACM SIGKDD Explor 8(1):93–104
78. Yan X, Cheng H, Han J, Xin D (2005) Summarizing itemset patterns: a profile-based approach, In: Proceedings of 11th ACM SIGKDD international conference on knowledge discovery and data mining. ACM Press, pp 314–323
79. Yang C, Fayyad U, Bradley P (2001) Efficient discovery of error-tolerant frequent itemsets in high dimensions, In: Proceedings of 7th ACM SIGKDD international conference on knowledge discovery and data mining. ACM Press, pp 194–203
80. Yin X, Han J (2003) CPAR: Classification based on predictive association rules. In: Proceedings of SIAM international conference on data mining
81. Zaki M (2004) Mining non-redundant association rules. Data Min Knowl Discov 9(3):223–248
82. Zaki M, Hsiao C-J (2002) CHARM: An efficient algorithm for closed itemset mining. In: Proceedings of 2nd SIAM international conference on data mining SIAM. pp 457–473
83. Zhao Y, Karypis G (2002) Evaluation of hierarchical clustering algorithms for document datasets. In: Proceedings of 11th Conference of information and knowledge management. pp 515–524

## Author Biographies

**Fabian Moerchen** graduated with a Ph.D. in Feb 2006 from the Philipps University of Marburg, Germany after just over 3 years with summa cum laude. In his thesis he proposed a radically different approach to temporal interval patterns that uses itemset and sequential pattern mining paradigms. Since 2006 he has been working at Siemens Corporate Research, a division of Siemens Corporation, leading data mining projects with applications in predictive maintenance, text mining, healthcare, and sustainable energy. He has continued the study of temporal data mining in the context of industrial and scientific problems and has served the community as a reviewer, organizer of workshops, and presenter of tutorials.

**Michael Thies** received his Master degree in Computer Science from Philipps University of Marburg, Germany. He is currently working as a self-employed software developer and data analyst. His research interests include data mining in general and temporal pattern mining in particular. He further participated in the MusicMiner project that deployed signal processing and data mining methods to groups recorded musical pieces by perceived similarity.

**Alfred Ultsch** received his Ph.D. Degree in Computer Science from ETH Zurich, Switzerland. He holds Master degrees in Computer Science from TU Munich, Germany and Purdue University West Lafayette Indiana, USA. He is currently a Professor in the Philipps University of Marburg, Germany in the field of Databionics. Prof. Ultsch has published in the areas of bio-inspired computing, databionics, knowledge discovery recognition, bioinformatics and financial analysis.