# Mining Efficiently Significant Classification Association Rules

Yanbo J. Wang[1], Qin Xin[2], and Frans Coenen[1]

[1] Department of Computer Science, University of Liverpool, Ashton Building, Ashton Street, Liverpool, L69 3BX, UK. Emails:{jwang,frans}@csc.liv.ac.uk
[2] Department of Informatics, University of Bergen, P.B.7800, N-5020 Bergen, Norway. Email: xin@ii.uib.no

## Overview

Classification Rule Mining (CRM) is a well-known Data Mining technique for the extraction of hidden Classification Rules (CRs) from a given database that is coupled with a set of pre-defined classes, the objective being to build a classifier to classify "unseen" data-records. One recent approach to CRM is to employ Association Rule Mining (ARM) techniques to identify the desired CRs, i.e. Classification Association Rule Mining (CARM). Although the advantages of accuracy and efficiency offered by CARM have been established in many papers, one major drawback is the large number of Classification Association Rules (CARs) that may be generated – up to a maximum of "$2^n - n - 1$" in the worst case, where $n$ represents the number of data-attributes in a database. However, there are only a limited number, say at most $\hat{k}$ in each class, of CARs that are required to distinguish between classes. The problem addressed in this chapter is how to efficiently identify the $\hat{k}$ such CARs. Having a CAR list that is generated from a given database, based on the well-established "Support-Confidence" framework, a rule weighting scheme is proposed in this chapter, which assigns a score to a CAR that evaluates how significantly this CAR contributes to a single pre-defined class. Consequtently a rule mining approach is presented, that addresses the above, that operates in time $O(k^2n^2)$ in its deterministic fashion, and $O(kn)$ in its randomised fashion, where $k$ represents the number of CARs in each class that are potentially significant to distinguish between classes and $k \geq \hat{k}$; as opposed to exponential time $O(2^n)$ – the time required in score computation to mine all $\hat{k}$ CARs in a "one-by-one" manner. The experimental results show good performance regarding the accuracy of classification when using the proposed rule weighting scheme with a suggested rule ordering mechanism, and evidence that the proposed rule mining approach performs well with respect to the efficiency of computation.

## 1 Introduction

Data Mining [29, 30] is a promising area of current research and development in Computer Science, which is attracting more and more attention from a wide range of different groups of people. Data Mining aims to extract various types of hidden and interesting knowledge (i.e., rules, patterns, regularities, customs, trends, etc.) from databases, where the volume of a collected database can be very large. In Data Mining, common types of mined knowledge include: Association Rules (ARs) [1], Classification Rules (CRs) [45], Classification Association Rules (CARs) [3], Prediction Rules (PRs) [28], Clustering Rules (CTRs) [42], Sequential Patterns (SPs) [52], Emerging Patterns (EPs) [21], etc.

An AR describes a co-occurring relationship between binary-valued data-attributes, expressed in the form of an "antecedent $\Rightarrow$ consequent" rule. Association Rule Mining (ARM) [2], with its wide range of applications, has been well-established in Data Mining in the past decade. It aims to identify all ARs in a given transaction-database $D_T$. One application of ARM is to define CRs, from a training-dataset $D_R$ that is coupled with a set of pre-defined classes $C = \{c_1, c_2, ..., c_{|C|-1}, c_{|C|}\}$, which can be used to classify the data-records in a test-dataset $D_E$. This kind of AR based CR is referred to as CAR. In general the process to build a classifier using identified CRs is called Classification Rule Mining (CRM) [45], which is another well-known Data Mining technique paralleling ARM. In CRM, a class-database $D_C$ is given as $D_R \cup D_E$, where $D_R$ and $D_E$ share the same data-attributes but the class-attribute (the last data-attribute in $D_R$) is "unseen" in $D_E$.

Classification Association Rule Mining (CARM) [3] is a recent CRM approach that builds an ARM based classifier using CARs, where these CARs are generated from a given transaction-training-dataset $D_{TR} \subset D_{TC}$, and $D_{TC}$ is a $D_C$ in a "transactional" manner. In [18], Coenen *et al.* suggest that results presented in [36] and [38] show that CARM seems to offer greater accuracy, in many cases, than other methods such as C4.5 [45]. However, one major drawback of this approach is the large number of CARs that might be generated – up to a maximum of "$2^n - n - 1$" in the worst case, where $n$ represents the number of data-attributes in $D_{TC}$. In [53], Yin and Han belive that there are only a limited number, say at most $\hat{k}$ in each class, of CARs that are required to distinguish between classes and should be thus used to make up a classifer. They suggest a value of 5 as an appropriate value for $\hat{k}$, and employ the *Laplace expected error estimate* [10] to estimate the acuracy of CARs. In [15] Coenen and Leng evaluated a number of alternative rule ordering and case satisfaction strategies, and conclude that for lower confidence thresholds (i.e., 50% to 75%) CSA (Confidence-Support-size_of_Antecedent) and *Laplace* ordering coupled with a "best first" case satisfaction mechanism can achieve better accuracy than comparable alternatives.

## 1.1 Contribution

Given a CAR list $\mathcal{R} = \{R_1, R_2, ..., R_{N-1}, R_N\}$ that is generated from a given $D_{TR}$ based on the well-established "Support-Confidence" framework, where $\mathcal{R}$ is presented using CSA ordering, and $N$ represents the size of $\mathcal{R}$ that can be as large as "$2^n - n - 1$", a rule weighting scheme is proposed in this chapter, which assigns a score to a CAR $R_j \in \mathcal{R}$ that represents how significantly $R_j$ contributes to a single class $c_i \in C$. An alternative rule ordering mechanism is consequently introduced, based on the proposed rule weighting scheme, that aims to improve the performance of the well-established CSA ordering regarding the accuracy of classification. In [19] a general framework for selectors, namely $(k, m, n)$-selectors, was proposed with applications in optimal group testing. In this chapter, a similar concept of selectors is further considered in a randomised setting. This randomised selector can be proved to exist with a high probability. With regards to the concept of selectors, a novel rule mining approach is presented that addresses the problem of mining the $\hat{k}$ "significant rules" (see Definition 9 in section 3.2) in $\mathcal{R}$. The rule mining approach operates in time $O(k^2n^2)$ in its deterministic fashion, and $O(kn)$ in its randomised fashion, where $k$ represents the number of CARs in each class that can potentially be used to distinguish between classes and $k \geq \hat{k}$; as opposed to exponential time $O(2^n)$ – the time required in score computation to find all $\hat{k}$ significant (the "best $\hat{k}$") rules in $\mathcal{R}$ in a "one-by-one" manner. The experimental results show that the proposed rule weighting and rule ordering approaches perform well regarding the accuracy of classification, and evidence the fast computational efficiency of running the randomised rule mining approach. Note that the deterministic rule mining approach is theoretical only.

## 1.2 Chapter Organisation

The following section describes some Data Mining work that relate to CARM. In section 3 we first introduce the rule weighting scheme together with a rule ordering mechanism based on the rule weighting scheme; and sketch the concept of deterministic selectors and give an introduction to the concept of randomised selectors. In section 4 we propose a rule mining approach to efficiently mine the "best $\hat{k}$" CARs in $\mathcal{R}$. In section 5 we present experimental results obtained using the TFPC (Total From Partial Classification) CARM algorithm [15, 18] coupled with a "best first" case satisfaction approach. Finally we discuss our conclusions in section 6, and a number of open issues for further research.

## 2 Related Work

### 2.1 Association Rule Mining

ARM extracts a set of ARs from $D_T$, first introduced in [1]. Let $I = \{a_1, a_2, ..., a_{n-1}, a_n\}$ be a set of items (data-attributes), and $\mathcal{T} = \{T_1, T_2, ..., T_{m-1}, T_m\}$ be a set of transactions (data-records), $D_T$ is described by $\mathcal{T}$, where each $T_i \in \mathcal{T}$ contains a set of items $I' \subseteq I$. In ARM, two threshold values are usually used to determine the significance of an AR:

- **Support:** The frequency that the items occur or co-occur in $\mathcal{T}$. A support threshold $\sigma$, defined by the user, is used to distinguish frequent items from the infrequent ones. A set of items $S$ is called an itemset, where $S \subseteq I$, and $\forall a_i \in S$ co-occur in $\mathcal{T}$. If the occurrences of some $S$ in $\mathcal{T}$ exceeds $\sigma$, we say that $S$ is a Frequent Itemset (FI).
- **Confidence:** Represents how "strongly" an itemset $X$ implies another itemset $Y$, where $X, Y \subseteq I$ and $X \cap Y = \{\oslash\}$. A confidence threshold $\alpha$, supplied by the user, is used to distinguish high confiderence ARs from low confidence ARs.

An AR $X \Rightarrow Y$ is valid when the support for the co-occurrence of $X$ and $Y$ exceeds $\sigma$, and the confidence of this AR exceeds $\alpha$. The computation of support is $\frac{X \cup Y}{|\mathcal{T}|}$, where $|\mathcal{T}|$ is the size function of the set $\mathcal{T}$. The computation of confidence is $\frac{Support(X \cup Y)}{Support(X)}$. Informally, $X \Rightarrow Y$ can be interpreted as "if $X$ exists, it is likely that $Y$ also exists". With regards to the history of ARM investigation, three major categories of serial (non-parallel) ARM algorithms can be identified: (1) mining ARs from all possible FIs, (2) mining ARs from Maximal Frequent Itemsets (MFIs), and (3) mining ARs from Frequent Closed Itemsets (FCIs).

#### 2.1.1 Mining ARs From FIs

In the past decade, many algorithms have been introduced that mine ARs from identified FIs. These algorithms can be further grouped into different "families", such as Pure-apriori like, Semi-apriori like, Set Enumeration Tree like, etc.

- **Pure-apriori like** where FIs are generated based on the generate-prune level by level iteration that was first promulgated in the Apriori algorithm [2]. In this "family" archetypal algorithms include: Apriori, AprioriTid and AprioriHyprid [2], Partition [49], DHP [43], Sampling [50], DIC [7], CARMA [31], etc.
- **Semi-apriori like** where FIs are generated by enumerating candidate itemsets but do not apply the Apriori generate-prune iterative approach founded on (1) the join procedure, and (2) the prune procedure that employs the closure property of itemsets – if an itemset is frequent then all

its subsets will also be frequent; if an itemset is infrequent then all its supersets will also be infrequent. In this "family" typical algorithms include: AIS [1], SETM [33], OCD [40], etc.

- **Set Enumeration Tree like** where FIs are generated through constructing a set enumeration tree structure [48] from $D_T$, which avoids the need to enumerate a large number of candidate itemsets. In this "family" a number of approaches can be further divided into two main streams: (1) Apriori-TFP[1] based (i.e., [11], [12], [13], [16], [17], etc.), and (2) FP-tree based (i.e., [9], [24], [27], [39], etc.).

### 2.1.2 Mining ARs From MFIs

It is apparent that the size of a complete set of FIs can be very large. The concept of MFI [47] was proposed to find several "long" (super) FIs in $D_T$, which avoids the redundant work required to identify "short" FI. The concept of vertical mining has also been effectively promoted in this category [54]. Vertical mining, first mentioned in [32], deals with a vertical transaction database $D_T^V$, where each data-record represents an item that is associated with a list of its relative transactions (the transactions in which it is present). Typical MFI algorithms include: MaxEclat / Eclat [54], MaxClique / Clique [54], Max-Miner [47], Pincer-Search [37], MAFIA [8], GenMax [26], etc.

### 2.1.3 Mining ARs From FCIs

Algorithms belonging to this category extract ARs through generating a set of FCIs from $D_T$. In fact the support of some sub-itemsets of an MFI might be hard to identified resulting in a further difficulty in the computation of confidence. The concept of FCI [44] is proposed to improve this property of MFI, which avoids the difficulty of identifying the support of any sub-itemsets of a relatively "long" FI. A FCI $f$ is an itemset $S \in D_T$, where $f$ is frequent and $\neg\exists$ itemset $f' \supset f$ and $f'$ shares a common support with $f$. The relationship between FI, MFI and FCI is that MFI $\subseteq$ FCI $\subseteq$ FI [8]. In this category typical algorithms include: CLOSET [44], CLOSET+ [51], CHARM [55], MAFIA [8], etc.

## 2.2 Classification Rule Mining

CRM deals with $D_C$, where $D_C$ is founded as $D_R \cup D_E$. It discovers a set of CRs in $D_R$ from which to build a classifier to classify "unseen" data records in $D_E$. A $D_R$ consists of $n$ data-attributes and $m$ data records. By convention the last data-attribute in each data-record usually indicates its pre-defined class, noted as the class attribute. CRM can thus be described as the process of

[1] Apriori-TFP and its related softwares may be obtained from http://www.csc.liv.ac.uk/~frans/KDD/Software.

assigning a Boolean value to each pair $(d_j, c_i) \in D_E \times C$, where each $d_j \in D_E$ is an "unseen" data-record, $C$ as declared in section 1 is a set of pre-defined classes, and $(d_j, c_i)$ is a data-record in $D_E$ to be labeled.

### 2.2.1 The Cover Algorithm

In CRM a number of approaches have been proposed to generate a classifier from a set of training data-records. For example the Cover Algorithm [41] takes $D_R$ as its input and aims to generate a complete set of minimal non-redundant CRs. We define the Cover algorithm in **fig. 1.** as follows.

```
Algorithm Cover;
input: $D_R$ (a training-dataset);
output: the set $\mathcal{S}_{\mathcal{CR}}$ (the complete set of minimal non-redundant CRs);
(1)begin
(2)    $\mathcal{S}_{\mathcal{CR}} := \{\oslash\}$;
(3)    while $D_R \neq \{\oslash\}$ do
(4)        find a CR cr from $D_R$ heuristically;
(5)        remove all records identified by cr from $D_R$;
(6)        $\mathcal{S}_{\mathcal{CR}} \leftarrow \mathcal{S}_{\mathcal{CR}} \cup cr$;
(7)    end while
(8)    return ($\mathcal{S}_{\mathcal{CR}}$);
(9)end
```

**Fig. 1.** The Cover Algorithm

### 2.2.2 Existing CRM Approaches

With regards to the history of CRM investigation, various mechanisms on which CRM algorithms have been based include: Decision Trees [45], Bayesian Approach [20], $K$-Nearest Neighbour [34], Support Vector Machine [6], Association Rules [38], Emerging Patterns [22], Genetic Algorithm [25], Neural Networks [28], Case-based Reasoning [28], Rough Set [28], Fuzzy Set [28], Simple Approach [23], etc. In this section, we briefly describe four of the most well-known mechanisms used in CR generation as follows.

- **Decision Trees:** Where CRs are mined based on a greedy algorithm. The approach can be separated into two stages where a flow chart like tree structure is constructed from $D_R$ first (stage 1) followed by a tree pruning phase; the pruned tree is then used in CR generation (stage 2). C4.5 [45] is the most famous Decision Tree based CRM method and operates by recursively splitting $D_R$ on the attribute that produces the *maximum gain* to generate the decission tree. This tree is then pruned according to an error estimate. The result is used to classify "unseen" data.

- **Bayesian Approach:** The typical mechanism found in Byesian CRM approaches is naive bayes, which has been widely applied in Machine Learning. The general idea of naive bayes is to make use of knowledge of the joint probabilities that exist between attributes in training-dataset so as to produce a model of some machine learning application that can be applied to "unseen" data. The term naive is used to refer to the assumption that the conditional probability between data-attributes is independent of the conditional probability between other data-attributes. A naive bayes classifier is built using $D_R$, which comprises a set of conditional probabilities for each data-attribute $a_h \in I_R$ (the set of attributes in $D_R$) and each class $c_i \in C$, so that there are $|I_R| \times |C|$ probabilities. This set of conditional probabilities is then used to classify "unseen" data-records in $D_E$.
- **$K$-Nearest Neighbour:** $K$-Nearest Neighbour ($K$-NN) is a well-known statistical approach used in CRM, which classifies an "unseen" data-record $d_{Ei} \in D_E$, by summarising a common pre-defined class from its $K$ most similar instances, identified in $D_R$. To identify the $K$ most similar training-instances for $d_{Ei}$, calculating the Euclidean distance value between each training data-record $d_{Ri} \in D_R$ and $d_{Ei}$ has been commonly used: $Distance(d_{Ri}, d_{Ei}) = \sqrt{(\sum_{j=1}^{n}(d_{Ri_j} - d_{Ei_j})^2)}$, where $d_{Ri_j}$ and $d_{Ei_j}$ are the values of the $j$th data-attribue in $D_C$ for $d_{Ri}$ and $d_{Ei}$.
- **Support Vector Machine:** The objective of using Support Vector Machine (SVM) [6] is to find a hypothesis $\tilde{h}$ which minimises the *true error* defined as the probability that $\tilde{h}$ produces an erroneous result. SVM make use of linear functions of the form: $f(x) = w\ ^T x + b$, where $w$ is the weight vector, $x$ is the input vector, and $w\ ^T x$ is the inner product between $w$ and $x$. The main concept of SVM is to select a *hyperplane* that separates the positive and negative examples while maximising the smallest margin. Standard SVM techniques produce binary classifiers. Two common approaches to support the application of SVM techniques to the multi-class problem are One Against All (OAA) and One Against One (OAO).

### 2.3 Classification Association Rule Mining

An overlap between ARM and CRM is CARM, which strategically solves the traditional CRM problem by applying ARM techniques. It mines a set of CARs from $D_{TR}$. A CAR is an AR of the form $X \Rightarrow c_i$, where $X$ is an FI mined from $D_{TR}$, and $c_i$ is a pre-defined class in $C$ to which data-records can be assigned. The idea of CARM was first presented in [3]. Subsequently a number of alternative approaches have been described. Broadly CARM algorithms can be categorised into two groups according to the way that the CRs are generated:

- **Two stage algorithms** where a set of CARs are produced first (stage 1), which are then pruned and placed into a classifier (stage 2). Examples of this approach include CBA [38] and CMAR [36]. CBA (Classification

Based on Associations), developed by Liu *et al.* in 1998, is an Apriori [2] based CARM algorithm, which (1) applies its CBA-GR procedure for CAR generation; and (2) applies its CBA-CB procedure to build a classifier based on the generated CARs. CMAR (Classification based on Multiple Association Rules), introduced by Han and Jan in 2001, is similar to CBA but generates CARs through a FP-tree [27] based approach.

- **Integrated algorithms** where the classifier is produced in a single processing step. Examples of this approach include TFPC[2] [15, 18], and induction systems such as FOIL [46], PRM and CPAR [53]. TFPC (Total From Partial Classification), proposed by Coenen *et al.* in 2004, is a Apriori-TFP [16] based CARM algorithm, which generates CARs through efficiently constructing both P-tree and T-tree set enumeration tree structures. FOIL (First Order Inductive Learner) is an inductive learning algorithm for generating CARs developed by Quinlan and Cameron-Jones in 1993. This algorithm was later developed by Yin and Han to produce the PRM (Predictive Rule Mining) CAR generation algorithm. PRM was then further developed, by Yin and Han in 2003 to produce CPAR (Classification based on Predictive Association Rules).

### 2.3.1 Case Satisfaction Approaches

Regardless of which particular methodlogy is used to build it, a classifier is ususally presented as an ordered CAR list $\mathcal{R}$. In [15] Coenen and Leng summarised three case satisfaction approaches that have been employed in different CARM algorithms for utilising the resulting classifier to classify "unseen" data. These three case satisfaction approaches are itemised as follows (given a particular case):

- **Best First Rule:** Select the first "best" rule that satisfies the given case according to some ordering imposed on $\mathcal{R}$. The ordering can be defined according to many different ordering schemes, including: (1) CSA (Confidence-Support-size_of_Antecedent) – combinations of confidence, support and size of antecedent, with confidence being the most siginificant factor (used in CBA, TFPC and the early stages of processing of CMAR); (2) WRA (Weighted Relative Accuracy) – which reflects a number of rule "interestingness" measures as proposed in [35]; (3) Laplace Accuracy – as used in PRM and CPAR; (4) $\chi^2$ Testing – $\chi^2$ values as used, in part, in CMAR; (5) ACS (size_of_Antecedent-Confidence-Support) – an alternative to CSA that considers the size of the rule antecedent as the most significant factor; etc.
- **Best $\mathcal{K}$ Rules:** Select the first "best $\mathcal{K}$" rules (in this chapter we denote $\mathcal{K}$ by $\hat{k}$ as mentioned above) that satisfy the given case and then select a rule according to some averaging process as used for example, in CPAR.

[2] TFPC may be obtained from http://www.csc.liv.ac.uk/~frans/KDD/Software.

The term "best" in this case is defined according to an imposed ordering of the form described in **Best First Rule**.

- **All Rules:** Collect all rules in the classifier that satisfy the given case and then evaluate this collection to identify a class. One well-known evaluation method in this category is WCS (Weighted $\chi^2$) testing as used in CMAR.

### 2.3.2 Rule Ordering Approaches

As noted in the previous section five existing rule ordering mechanisms are identified to support the "best first rule" case satisfaction strategy. Each can be further separated into two stages: (1) a rule weighting stage where each $R_j \in \mathcal{R}$ is labeled with a weighting score that represents the significance of $R_j$ indicates a single class $c_i$; and (2) a rule re-ordering stage, which sorts the original $\mathcal{R}$ in a descending manner, based on the score assigned in stage (1), of each $R_j$. With regards to both stages of rule weighting and rule re-ordering, each rule ordering mechanism can be described in more detail as follows:

- **CSA:** The CSA rule ordering mechanism is based on the well-established "Support-Confidence" framework (see section 2.1). It does not assign an additional weighting score to each $R_j \in \mathcal{R}$ in its rule weighting stage, but simply gathers the values of confidence and support, and the size of the rule antecedent to "express" a weighting score for each $R_j \in \mathcal{R}$. In the rule re-ordering stage, CSA generally sorts the original $\mathcal{R}$ in a descending order based on the value of confidence of each $R_j$. For these rules in $\mathcal{R}$ that share a common value of confidence, CSA sorts them in a descending order based on their support value. Furthermore for these rules in $\mathcal{R}$ that share common values for both confidence and support, CSA sorts them in an ascending order based on their size of the rule antecedent.
- **WRA:** The use of WRA can be found in [35], where this technique is used to determine an expected accuracy for each generated CR. In its rule weighting stage, WRA assigns a weighting score to each $R_j \in \mathcal{R}$. The calculation of the value of $R_j$, confirmed in [15], is: $\mathit{wra}(R_j) = \mathit{support}(R_j.\mathit{antecedent}) \times (\mathit{confidence}(R_j)$ - $\mathit{support}(R_j.\mathit{consequent})$. In the rule re-ordering stage the origninal $\mathcal{R}$ is simply sorted in a descending order based on the assigned *wra* value of each $R_j$.
- **Laplace Accuracy:** The use of the *Laplace expected error estimate* [10] can be found in [53]. The principle of applying this rule ordering mechanism is similar to WRA. The calculation of the *Laplace* value of $R_j$ is: $\mathit{Laplace}(R_j) = \frac{\mathit{support}(R_j.\mathit{antecedent} \cup R_j.\mathit{consequent}) + 1}{\mathit{support}(R_j.\mathit{antecedent} + |C|)}$, where $|C|$ is the size function of the set $C$.
- **$\chi^2$ Testing:** $\chi^2$ Testing is a well known technique in statistics, which can be used to determine whether two variables are independent of one another. In $\chi^2$ Testing a set of observed values ($O$) is compared against a set of expected values ($E$) – values that would be estimated if there were no associative relationship between the variables. The value of $\chi^2$ is calculated

as: $\sum_{i=1}^{n} \frac{(O_i - E_i)^2}{E_i}$, where $n$ is the number of observed/expected values, which is always 4 in CARM. If the $\chi^2$ value between two variables (the antecedent and consequent of $R_j \in \mathcal{R}$) above a given threshold value (for CMAR the chosen threshold is 3.8415), thus it can be concluded that there is a relation between the rule antecedent and consequent, otherwise there is not a relation. After assigning a $\chi^2$ value to each $R_j \in \mathcal{R}$, it can be used to re-order the $\mathcal{R}$ in a descending basis.

- **ACS:** The ACS rule ordering mechanism is a variation of CSA. It takes the size of the rule antecedent as its major factor (using a descending order) followed by the rule confidence and support values respectively. This rule ordering mechanism ensures that "specific rules have a higher precedence than more general rules" [15].

## 3 Preliminaries

As noted in section 1.1, $\mathcal{R} = \{R_1, R_2, \cdots, R_{2^n-n-2}, R_{2^n-n-1}\}$ represents a complete set of possible CARs that are generated from $D_{TR}$, and $R_j$ represents a rule in set $\mathcal{R}$ with label $j$.

### 3.1 Proposed Rule Weighting Scheme

#### 3.1.1 Item Weighting Score

There are $n$ items involved in $D_{TR}$. For a particular pre-defined class $A$ (as $c_i \in C$), a score is assigned to each item in $D_{TR}$ that distingushes the significant items for class $A$ from the insignificant ones.

**Definition 1.** *Let $\c{c}^A(Item_h)$ denote the contribution of each $item_h \in D_{TR}$ for class $A$, which represents how significantly $item_h$ determines $A$, where $0 \leq \c{c}^A(Item_h) \leq |C|$, and $|C|$ is the size function of the set $C$.*

The calculation of $\c{c}^A(Item_h)$ is given as follows:

$$\c{c}^A(Item_h) = (TransFreq(Item_h, A)) \times (1 - TransFreq(Item_h, \bar{A})) \times \frac{|C|}{ClassCount(Item_h,\ C)},$$

where
(1) The $TransFreq(Item_h,\ A\ or\ \ \bar{A})$ function computes how frequently that $Item_h$ appears in class $A$ or the group of classes $\bar{A}$ (the complement of $A$). The calculation of this function is: $\frac{number\ of\ transactions\ with\ Item_h\ in\ the\ class(es)}{number\ of\ transactions\ in\ the\ class(es)}$;

(2) The $ClassCount(Item_h, C)$ function simply counts the number of classes in $C$ which contain $Item_h$.

The rationale of this item weighting score is demonstrated as follows:

1. The weighting score of $Item_h$ for class $A$ tends to be high if $Item_h$ is frequent in $A$;
2. The weighting score of $Item_h$ for class $A$ tends to be high if $Item_h$ is infrequent in $\bar{A}$;
3. The weighting score of $Item_h$ for any class tends to be high if $Item_h$ is involved in a small number of classes in $C$. In [5], a similar idea can be found in feature selection for text categorisation.

### 3.1.2 Rule Weighting Score

Based on the item weighting score, a weighting score is assigned to the rule antecedent of each $R_j \in \mathcal{R}$.

**Definition 2.** *Let $\varsigma^A(R_j)$ denote the contribution of each CAR $R_j \in \mathcal{R}$ for class $A$, which represents how significantly $R_j$ determines $A$.*

The calculation of $\varsigma^A(R_j)$ is given as follows:

$$\varsigma^A(R_j) = \sum_{h=1}^{|R_j|} \varsigma^A(Item_h \in R_j) \ .$$

## 3.2 Some Definitions

**Definition 3.** *If $\varsigma^A(Item_h) < \varepsilon$, we recognise $Item_h \in D_{TR}$ as a light-weighted item for class $A$, where $\varepsilon$ is a very small user-defined constant and $0 \leq \varepsilon \leq 1$. We use $I'(A) = \{Item'_1, Item'_2, \cdots, Item'_{|I'(A)|-1}, Item'_{|I'(A)|}\}$ to denote the sufficient set of light-weighted items for $A$, identified in $D_{TR}$.*

**Definition 4.** *If a CAR $R_j \in \mathcal{R}$ significantly satisfies the following inequality, $\varsigma^A(R_j) > \sum_{h=1}^{|I'(A)|} \varsigma^A(Item_h \in I'(A))$, where the contribution of $R_j$ for class $A$ is significantly greater than the sum of the contributions of all light-weighted items for class $A$, we recognise $R_j$ as a heavy-weighted rule for $A$. We use $\mathcal{R}'(A) = \{R'_1, R'_2, \cdots, R'_{t-1}, R'_t\}$ to denote the set of selected heavy-weighted rules for $A$, identified in $\mathcal{R}$. In $\mathcal{R}$, we always select the top-$t$ heavy-weighted rules to construct $\mathcal{R}'(A)$, where integer $t$ is a user-defined small constant.*

**Definition 5.** *We recognise an item $Item_h \in D_{TR}$ as a heavy-weighted rule-item for class $A$ if $Item_h \in (\exists R'_j \in \mathcal{R}'(A))$. We use $I''(A) = \{Item''_1, Item''_2, \cdots, Item''_{|I''(A)|-1}, Item''_{|I''(A)|}\}$ to denote the sufficient set of heavy-weighted rule-items for $A$, identified in $D_{TR}$.*

**Definition 6.** *If a CAR $R_j \in \mathcal{R}$ does not contain any item $Item''_h \in I''(A)$, we recognise $R_j$ as a noisy rule for class $A$. We use $\mathcal{R}''(A) = \{R''_1, R''_2, \cdots, R''_{k'-1}, R''_{k'}\}$ to denote the sufficient set of noisy rules for $A$, identified in $\mathcal{R}$, where integer $k'$ represents the size of $\mathcal{R}''(A)$.*

**Definition 7.** *We recognise an item $Item_h \in D_{TR}$ as a noisy rule-item for class $A$ if $Item_h \in (\exists R''_j \in \mathcal{R}''(A))$. We use $I'''(A) = \{Item'''_1, Item'''_2, \cdots, Item'''_{|I'''(A)|-1}, Item'''_{|I'''(A)|}\}$ to denote the sufficient set of noisy rule-items for $A$, identified in $D_{TR}$.*

**Definition 8.** *If a CAR $(R_j \in \mathcal{R}) \notin \mathcal{R}''(A)$, we recognise $R_j$ as a potential significant rule for class $A$. We use $\mathcal{R}'''(A) = \{R'''_1, R'''_2, \cdots, \; R'''_{k-1}, R'''_k\}$ to denote the sufficient set of potential significant rules for class $A$, identified in $\mathcal{R}$ as $\mathcal{R} - \mathcal{R}''(A)$, where $k \geq t$ (See Definition 4).*

**Definition 9.** *If a CAR $(R_j \in \mathcal{R}) \in \mathcal{R}'''(A)$ satisfies the following inequality, $\c{c}^A(R_j) > \sum_{h=1}^{|I'''(A)|} \c{c}^A(Item'''_h \in I'''(A))$, where the contribution of $R_j$ to class $A$ is greater than the sum of the contributions of all noisy rule-items for class $A$, we recognise $R_j$ as a significant rule for $A$. We say there are at most $\hat{k}$ significant rules in $\mathcal{R}$, where integer $\hat{k}$ is a small user-defined constant $\leq k$.*

## 3.3 Proposed Rule Ordering Mechanism

In section 2.3.2 five existing rule ordering strategies were presented. Each is separated into both rule weighting and rule re-ordering stages. From the previous section, a list of CARs $\mathcal{R}^\diamond \subset \mathcal{R}$ has been generated that only consists of the "best $\hat{k}$" rules for each class $c_i \in C$, identified in $\mathcal{R}$. A rule re-ordering strategy is then required in the process of rule ordering. The rule re-ordering mechanism proposed herein contains three steps as follows:

1. $\mathcal{R}^\diamond$ is ordered using the well-established CSA ordering strategy;
2. The original $\mathcal{R}$ is linked at back of $\mathcal{R}^\diamond$, as $\mathcal{R}^\diamond + \mathcal{R}$;
3. Reassign: $\mathcal{R} \leftarrow (\mathcal{R}^\diamond + \mathcal{R})$.

## 3.4 Deterministic Selectors

We say that a set $P$ hits a set $Q$ on element $q$, if $P \cap Q = \{q\}$, and a family $\mathcal{F}$ of sets hits a set $Q$ on element $q$, if $P \cap Q = \{q\}$ for at least one $P \in \mathcal{F}$. De Bonis *et al.* [19] introduced a definition of a family of subsets of set $[N] \equiv \{0, 1, \cdots, N-2, N-1\}$ which hits each subset of $[N]$ of size at most $k$ on at least $m$ distinct elements, where $N, k$ and $m$ are parameters, $N \geq k \geq m \geq 1$. They proved the existence of such a family of size $O((k^2/(k-m+1)) \log N)$. For convenience of our presentation, we prefer the following slight modification of this definition, obtained by using the parameter $r = k - m$ instead of the parameter $m$. For integers $N$ and $k$, and a real number $r$ such that $N \geq k \geq r \geq 0$, a family $\mathcal{F}$ of subsets of $[N]$ is a $(N, k, r)$-selector, if for any subset $Q \subset [N]$ of size at most $k$, the number of all elements $q$ of $Q$ such that $\mathcal{F}$ does not hit $Q$ on $q$ is at most $r$. That is,

$$|\{q \in Q : \forall P \in \mathcal{F}, P \cap Q \neq \{q\}\}| \leq r \; .$$

In terms of this definition, De Bonis *et al.* [19] showed the existence of a $(N, k, r)$-selector of size $T(N, k, r) = O((k^2/(r+1)) \log N)$. In particular, there exists a $(N, k, 0)$-selector of size $O(k^2 \log N)$ such a “strong” selector hits each set $Q \subset [N]$ of size at most $k$ on each of its elements.

### 3.5 Proposed Randomised Selectors

A randomised $k$-selector $\mathcal{F}$ is a family of subsets of set $[N] \equiv \{0, 1, \cdots, N-2, N-1\}$ which hits each element $q$ of the subset $Q \subset [N]$ of size at most $k$ with high probability.

**Theorem 1.** *There exists a randomised $k$-selector $\mathcal{F}$ of size $O(k)$ such that $\mathcal{F}$ hits each set $Q \subset [N]$ of size at most $k$ on each of its elements with constant probability $p \geq 1/8$.*

*Proof.* Let each element $v \in [N]$ with uniformed probability $1/k$ to be the part of the element of $\mathcal{F}$. Let $H$ denote the number of different elements in $Q \subset [N]$ that have been hit by $\mathcal{F}$ after repeating the same procedure $k$ times. The probability $p$ that $\mathcal{F}$ hits each set $Q$ of size at most $k$ on each of its elements could be bounded by

$$\begin{aligned}
p &= E(H)/E(k) \\
&= \{\textstyle\sum_{i=1}^{k} [(k-i+1)/k] \times k \times (1/k) \times (1-1/k)^{k-1}\}/k \\
&= \textstyle\sum_{i=1}^{k} \{[(k-i+1)/k^2] \times (1-1/k)^{k-1}\} \\
&> \textstyle\sum_{i=1}^{k} \{[(k-i+1)/k^2] \times (1-1/k)^{k}\} \\
&> \textstyle\sum_{i=1}^{k} \{[(k-i+1)/k^2] \times (1/4)\} \qquad\qquad (\star) \\
&= (1/4) \times [(1+k) \times (k/2)/k^2] \\
&= (1/4) \times [(1+k)/(2k)] \\
&> (1/4) \times [k/(2k)] \\
&> 1/8 \, ,
\end{aligned}$$

where inequality $(\star)$ follows from the fact that the sequence $(1 - 1/k)^k$ is monotonely increasing.

## 4 Proposed Rule Mining Approach

### 4.1 The Strategy of the Deterministic Approach

To identify the significant CARs in $\mathcal{R}$, we provide a deterministic approach that employs a single application of a “strong” $(2^n, k, 0)$-selector. This approach ensures that every potential significant rule in $\mathcal{R}$ will be hit at least once. To apply a family $\mathcal{F}$ of subsets of $[2^n]$ means first to arrange the sets of $\mathcal{F}$ into a sequence $F_1, F_2, \cdots, F_{|\mathcal{F}|}$. Then in the $i$th step, only CARs in $\mathcal{R}$ with labels in $F_i$ will be involved in the procedure SIGNIFICANCE-TEST, while other CARs can be ignored. Thus, we have an $O(k^2 \log 2^n)$-complexity to hit each of the $k$ potential significant rules independently at least once, due to the property of the "strong" selector. If the current test for $F_i$ contributes to class $A$ significantly, then we call the function LOG-TEST, which is based on a binary search and finally finds one particular potential significant rule from $\mathcal{R}$ with labels in $F_i$. With a "smaller" list of rules (potential significant rules and some relevant potential significant rules), we then compute the weighting score of each rule, and finally catch the "best $\hat{k}$" (top-$\hat{k}$ score) rules.

## 4.2 The Strategy of the Randomised Approach

In this section, we use the randomised $k$-selector to substitute the "strong" selector in section 4.1. This randomised approach ensures that every potential significant rule in $\mathcal{R}$ will be hit at least once with high probability. To apply a family $\mathcal{F}$ of subsets of $[2^n]$ means first to arrange the sets of $\mathcal{F}$ into a sequence $F_1, F_2, \cdots, F_{|\mathcal{F}|}$. In the $i$th step, each element of $[2^n]$ will be contained in $F_i$ with probability $1/k$ and only CARs in $\mathcal{R}$ with labels in $F_i$ will be involved in the procedure SIGNIFICANCE-TEST, while other CARs can be ignored. Thus, we have an $O(k)$-complexity to hit each of the $k$ potential significant rules independently once with high probability, due to the property of the randomised $k$-selector (see Theorem 1). With a list of the extracted $k$ potential significant rules, the weighting score of each rule is computed. The "best $\hat{k}$" (top-$\hat{k}$ score) rules are the significant rules identified in $\mathcal{R}$.

## 4.3 Rule Mining Algorithm

The following function (**Fig. 2.**) identifies a potential significant rule in $\mathcal{R}$.

**Lemma 1.** *If there exists only one potential significant rule $R_j$ in the current test $F_i$ and $R_j$ is also a significant rule, then Function* LOG-TEST *will figure out $R_j$.*

*Proof.* According to Definition 9, we know that $ç^A(R_j) > \sum_{h=1}^{|I'''(A)|} ç^A(Item'''_h \in I'''(A))$. Thus, the subset of $\mathcal{R}$ which contains $R_j$ is always chosen for further binary test if $R_j$ is the only potential significant rule including in the current test $F_i$.

The following procedure (**Fig. 3.**) identifies all $\hat{k}$ significant rules in $\mathcal{R}$.

```
Function LOG-TEST($F_i$, $\mathcal{R}$);
input: $F_i$ (the $i$th element in $\mathcal{F}$) and set $\mathcal{R}$;
output: $Rw$ (a potential significant rule in $\mathcal{R}$);
(1)begin
(2)    $Rw := null$;
(3)    $Temp := F_i$;
(4)    while $|Temp| > 1$ do
(5)        choose an arbitrary subset $Temp_0$ with half CARs in $Temp$ to test;
(6)        if $\sum_{\forall Item_h \in Temp_0} \varsigma^A(Item_h) \geq \sum_{\forall Item_{h'} \in Temp - Temp_0} \varsigma^A(Item_{h'})$
(7)            then $Temp \leftarrow Temp_0$;
(8)            else $Temp \leftarrow Temp - Temp_0$;
(9)    end while
(10)   $Rw \leftarrow Temp$;
(11)   return ($Rw$);
(12)end
```

**Fig. 2.** The LOG-TEST Function

```
Procedure SIGNIFICANCE-TEST;
input: $\mathcal{F}$ ($(2^n, k, 0)$-selector / randomised $k$-selector for $[2^n]$), set $\mathcal{R}$, and integer $\hat{k}$;
output: the set $\mathcal{SR}$ (the set of significant rules);
(1)begin
(2)    $\mathcal{SR} := \{\oslash\}$;
(3)    $\mathcal{PSR} := \{\oslash\}$;
(4)    for $i = 1$ to $|\mathcal{F}|$ do
(5)        if the label of a CAR $R_j$ in $F_i$
(6)            then $R_j$ will be involved in current test;
(7)            else $R_j$ will be ignored in current test;
(8)        $\mathcal{PSR} \leftarrow \mathcal{PSR} \cup \{$LOG-TEST$(F_i, \mathcal{R})\}$;
(9)    end for
(10)   $\mathcal{SR} \leftarrow$ catch the top-$\hat{k}$ rules $R_j \in \mathcal{PSR}$ (according to $\varsigma^A(R_j)$);
(11)   return ($\mathcal{SR}$);
(12)end
```

**Fig. 3.** The SIGNIFICANCE-TEST Procedure

**Theorem 2.** *The Procedure* SIGNIFICANCE-TEST *will catch all $\hat{k}$ significant rules.*

*Proof.* According to the properties of the "selectors" (both deterministic and randomised selectors), we know that the selector $\mathcal{F}$ hits all $k$ potential significant rules at least once. Note that a significant rule is also a potential significant rule. Lemma 1 states that if current test $F_i$ hits only one significant rule $R_j$, then Function LOG-TEST will figure out $R_j$, which completes the proof of the theorem.

**Lemma 2.** *A $(2^n, k, 0)$-selector has size at most $O(k^2 n)$.*

*Proof.* It directly comes from the property of the selectors.

**Theorem 3.** *The problem of mining $\hat{k}$ significant rules in $\mathcal{R}$ can be solved in time $O(k^2n^2)$ in a deterministic manner, where $k$ is the number of potential significant rules in $\mathcal{R}$.*

*Proof.* Function LOG-TEST takes at most $\log 2^n$ time to find a pontential significant rule from a subset of $\mathcal{R}$. From Lemma 2, we know that a $(2^n, k, 0)$-selector has the size at most $O(k^2n)$. Consequently, the amount of time spent to figure out at most $k$ potential significant rules can be bounded by $O(k^2n^2)$. Finding the top-$\hat{k}$ significant rules in at most $O(k^2n)$ rules can be solved in time $O(k^2n\log(k^2n) + \hat{k})$, which is $O(k^2n^2)$ due to $\hat{k} \leq k \leq n$.

**Theorem 4.** *The problem of mining $\hat{k}$ potential significant rules in $\mathcal{R}$ can be solved in time $O(kn)$ in a randomised manner, with constant probability $p \geq 1/8$, where $k$ is the number of potential significant rules in $\mathcal{R}$.*

*Proof.* Function LOG-TEST takes at most $\log 2^n$ time to find a potential significant rule from a subset of $\mathcal{R}$. From Theorem 1, we know that a randomised $k$-selector has the size $O(k)$ with high probability to succeed. Consequently, the amount of time spent to figure out at most $k$ potential significant rules can be bounded by $O(kn)$ with constant probability $p \geq 1/8$. Finding the top-$\hat{k}$ significant rules in at most $O(k)$ rules can be solved in time $O(k\log(k) + \hat{k})$, which is $O(kn)$ due to $\hat{k} \leq k \leq n$.

## 5 Experimental Results

In this section, we aim to evaluate: (1) the proposed rule weighting and rule ordering strategies with respect to the accuracy of classification, where significant rules are mined in both (i) "one-by-one" or deterministic manner and (ii) randomised manner; and (2) the proposed rule mining approach (in its randomised fashion) with respect to the efficiency of computation by comparing it with the "one-by-one" mining approach. All evaluations were obtained using the TFPC CARM algorithm coupled with the "best first" case satisfaction strategy, although any other CARM classifier generator, founded on the "best first" strategy, could equally well be used. Experiments were run on a 1.20 GHz Intel Celeron CPU with 256 Mbyte of RAM runing under Windows Command Processor.

The experiments were conducted using a range of datasets taken from the LUCS-KDD discretised/normalised ARM and CARM Data Library [14]. The chosen datasets are originally taken from the UCI Machine Learning Repository [4]. These datasets have been discrelised and normalised using the LUCS-KDD DN software[3], so that data are then presented in a binary format uitable for use with CARM applications. It should be noted that the

[3] The LUCS-KDD DN is available at http://www.csc.liv.ac.uk/~frans/KDD/Software/LUCS-KDD-DN/.

datasets were rearranged so that occurences of classes were distributed evenly throughout the datasets. This then allowed the datasets to be divided in half with the first half used as the training-dataset and the second half as the test-dataset. Although a "better" accuracy figure might have been obtained using Ten-Cross Validation [25], it is the relative accuracy that is of interest here and not the absolute accuracy.

The first set of evaluations undertaken used a confidence threshold value of 50% and a support threshold value of 1% (as used in the published evaluations of CMAR [28], CPAR [53], TFPC [15, 18]). The results are presented in Table 1 where 120 classification accuracy values are listed based on 24 chosen datasets. The row labels describe the key characteristics of each dataset: for example, the label *adult.D97.N48842.C2* denotes the "adult" dataset, which includes 48,842 records in 2 pre-defined classes, with attributes that for the experiments described here have been discretised and normalised into 97 binary categories.

| *Datasets* | *CSA* | *One-by-one Approach* | | *Randomised Selector* | |
|---|---|---|---|---|---|
| | | $\hat{k} = 1$ | $\hat{k} = 10$ | $\hat{k} = 1$ $k = 5$ | $\hat{k} = 10$ $k = 50$ |
| adult.D97.N48842.C2 | 80.83 | 83.87 | 76.88 | 81.95 | 81.85 |
| anneal.D73.N898.C6 | 91.09 | 89.31 | 91.09 | 90.20 | 91.31 |
| auto.D137.N205.C7 | 61.76 | 64.71 | 59.80 | 64.71 | 58.82 |
| breast.D20.N699.C2 | 89.11 | 87.68 | 89.11 | 90.83 | 92.55 |
| connect4.D129.N67557.C3 | 65.83 | 66.78 | 65.87 | 66.34 | 66.05 |
| cylBands.D124.N540.C2 | 65.93 | 69.63 | 63.70 | 67.41 | 67.78 |
| flare.D39.N1389.C9 | 84.44 | 84.01 | 84.29 | 84.44 | 84.29 |
| glass.D48.N214.C7 | 58.88 | 64.49 | 52.34 | 64.49 | 64.49 |
| heart.D52.N303.C5 | 58.28 | 58.28 | 56.29 | 60.26 | 59.60 |
| hepatitis.D56.N155.C2 | 68.83 | 68.83 | 66.23 | 75.32 | 72.72 |
| horseColic.D85.N368.C2 | 72.83 | 77.72 | 80.43 | 80.43 | 81.52 |
| ionosphere.D157.N351.C2 | 85.14 | 84.00 | 90.29 | 88.57 | 93.14 |
| iris.D19.N150.C3 | 97.33 | 97.33 | 97.33 | 97.33 | 97.33 |
| led7.D24.N3200.C10 | 68.38 | 62.94 | 68.38 | 68.89 | 69.94 |
| letRecog.D106.N20000.C26 | 30.29 | 29.41 | 31.19 | 29.36 | 30.92 |
| mushroom.D90.N8124.C2 | 99.21 | 98.45 | 98.82 | 99.21 | 98.45 |
| nursery.D32.N12960.C5 | 80.35 | 76.85 | 76.17 | 80.20 | 81.11 |
| pageBlocks.D46.N5473.C5 | 90.97 | 91.74 | 90.97 | 91.74 | 90.97 |
| pima.D38.N768.C2 | 73.18 | 73.18 | 73.18 | 73.44 | 73.44 |
| soybean-large.D118.N683.C19 | 85.92 | 81.23 | 86.51 | 84.46 | 84.75 |
| ticTacToe.D29.N958.C2 | 71.61 | 68.48 | 72.03 | 71.19 | 73.28 |
| waveform.D101.N5000.C3 | 61.60 | 58.92 | 55.96 | 59.52 | 57.20 |
| wine.D68.N178.C3 | 53.93 | 83.15 | 71.91 | 83.15 | 85.39 |
| zoo.D42.N101.C7 | 76.00 | 86.00 | 78.00 | 90.00 | 86.00 |
| Average | 73.82 | 75.29 | 74.03 | 76.81 | 76.79 |

**Table 1** *Classification accuracy ($\alpha$ = 50%, $\sigma$ = 1%)*

From Table 1 it can be seen that with a 50% confidence threshold and an 1% support threshold the proposed rule weighting and rule ordering mechanisms worked reasonably well. When choosing a value of 1 as the value for

$\hat{k}$ (only the most significantly CAR for each class is mined) and applying the "one-by-one" rule mining approach, the average accuracy of classification throughout the 24 datasets is 75.29%. When substituting the value of 1 by a value of 10 (the best 10 significant CARs for each class are identified), the average accuracy, using the "one-by-one" rule mining approach, is 74.03%. Note that the average accuracies are higher than the average accuracy of classification obtained by the well-established CSA ordering appraoch, which is 73.82%. Furthermore when dealing with the randomised selector based rule mining approach, and choosing a value of 1 as the value for $\hat{k}$ and a value of 5 as the value for $k$ (only the most significantly CAR for each class is mined, based on the existance of 5 potential significant CARs for each class in $\mathcal{R}$), the average accuracy throughout the 24 datasets can be obtained as 76.81%. Note that in the randomised experiment process, we always run several tests (i.e., 8-10 tests) for each dataset, and catch the best result. When substituting the value of 1 by a value of 10, and the value of 5 by a value of 50 (the best 10 significant CARs for each class are mined, based on the existance of 50 potential significant CARs for each class in $\mathcal{R}$), the average accuracy was found as 76.79%.

| *Dataset* | *One-by-one Approach* | | | | *Randomised Selector* | | | |
|---|---|---|---|---|---|---|---|---|
| letRecog | $\hat{k} = 1$ | | | | $\hat{k} = 1$, $k = 50$ | | | |
| D106. N20000. C26 | Rule Number (before) | Rule Number (after) | Time (seconds) | Accuracy (%) | Rule Number (before) | Rule Number (after) | Time (seconds) | Accuracy (%) |
| 1 | 149 | 167 | 0.080 | 29.41 | 149 | 166 | 0.160 | 29.60 |
| 0.75 | 194 | 212 | 0.110 | 29.94 | 194 | 211 | 0.160 | 29.92 |
| 0.50 | 391 | 415 | 0.200 | 35.67 | 391 | 411 | 0.251 | 35.78 |
| 0.25 | 1118 | 1143 | 1.052 | 40.36 | 1118 | 1139 | 0.641 | 41.26 |
| 0.10 | 2992 | 3018 | 4.186 | 44.95 | 2992 | 3016 | 0.722 | 45.18 |
| 0.09 | 3258 | 3284 | 4.617 | 45.21 | 3258 | 3282 | 1.913 | 45.42 |
| 0.08 | 3630 | 3656 | 6.330 | 45.88 | 3630 | 3655 | 2.183 | 45.43 |
| 0.07 | 3630 | 3656 | 6.360 | 45.88 | 3630 | 3656 | 2.163 | 46.02 |
| 0.06 | 4366 | 4392 | 5.669 | 46.70 | 4366 | 4391 | 2.754 | 46.45 |
| 0.05 | 4897 | 4923 | 7.461 | 47.28 | 4897 | 4922 | 3.235 | 47.65 |
| 0.04 | 5516 | 5542 | 9.745 | 47.67 | 5516 | 5542 | 3.526 | 47.53 |
| 0.03 | 6341 | 6367 | 12.339 | 48.22 | 6341 | 6365 | 4.296 | 48.79 |

**Table 2** *Computational efficiency & Classification accuracy ($\alpha$ = 50%)*

The second set of evaluations undertaken used a confidence threshold value of 50%, a set of decreasing support threshold values from 1% to 0.03%, and the letter recognition dataset. The "large" letter recognition dataset (*letRecog.D106.N20000.C26*), comprises 20,000 records & 26 pre-defined classes. For the experiment the dataset has been discretised and normalised into 106 binary categories. From the experiment it can be seen that a relationship exists between: the selected value of support threshold ($\sigma$ or *min.support*), the number of generated CARs ($|\mathcal{R}|$), the accuracy of classification ($Accy$), and the time in seconds spent on mining significant rules ($Time$). Clearly, $\downarrow \sigma \Rightarrow \uparrow |\mathcal{R}| \Rightarrow (\uparrow Accy \wedge \uparrow Time)$.

Table 2 demonstrate that with a 50% confidence threshold and a value of 1 as the value for $\hat{k}$ (only the most significantly CAR for each class is mined in $|\mathcal{R}|$), the proposed rule mining approach (its randomised fashion) performs well with respect to both accuracy of classification and efficiency of computation. When applying the "one-by-one" rule mining approach, as $\sigma$ decreasing from 1% to 0.03%, $|\mathcal{R}|$ (before mining the "best $\hat{k}$" rules) is increased from 149 to 6,341; and $|\mathcal{R}|$ (after mining the "best $\hat{k}$" rules and re-ordering all rules) is increased from 167 to 6,367. Consequently accuracy has been increased from 29.41% to 48.22%, and $Time$ (the time spent in mining the $\hat{k}$ significant rules) has been increased from 0.08 seconds to 12.339 seconds. In comparison when applying the proposed randomised rule mining approach with a value of 50 as the value for $k$ (there exist 50 potential significant rules for each class in $|\mathcal{R}|$), as $\sigma$ decreasing from 1% to 0.03%, $|\mathcal{R}|$ (before mining the "best $\hat{k}$" rules) is increased from 149 to 6,341; and $|\mathcal{R}|$ (after mining the "best $\hat{k}$" rules and re-ordering all rules) is increased from 166 to 6,365. Consequently accuracy has been increased from 29.60% to 48.79%, and $Time$ (the time spent in mining the $\hat{k}$ significant rules) has been increased from 0.16 seconds to 4.296 seconds.

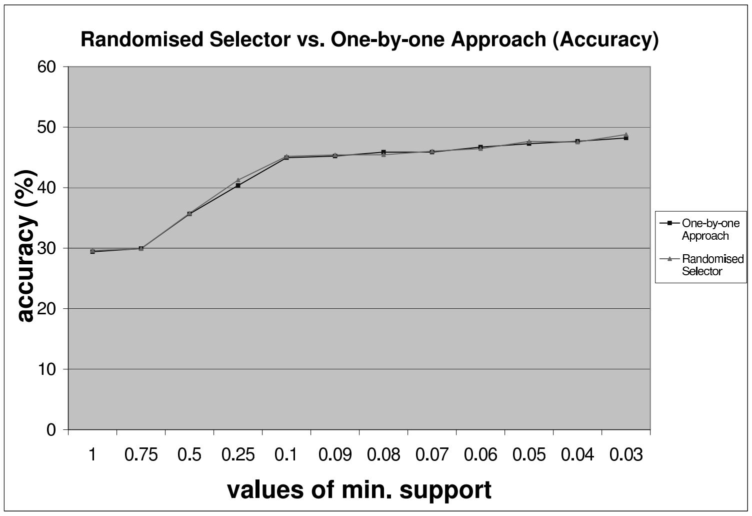

**Fig. 4.** Randomised Selector vs. One-by-one Approach (Accuracy)

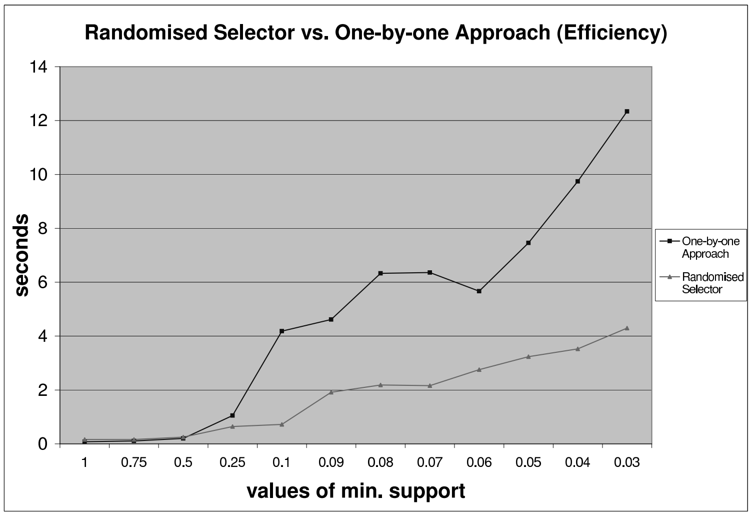

**Fig. 5.** Randomised Selector vs. One-by-one Approach (Efficiency)

**Fig. 4. & Fig. 5.** demonstrate (respectively) that there is no significant difference between accuracies of classification obtained by the "one-by-one" rule mining approach and the randomised selector based rule mining approach, whereas a significant difference in times spent on mining significant rules can be seen.

## 6 Conclusion

This chapter is concerned with an investigation of CARM. A overview of existing CARM algorithms was provided in section 2 where five existing rule weighting schemes used in CARM algorithms were reviewed. A rule weighting scheme was proposed in section 3 that ws used to distinguish the significant CARs from the insignificant ones. Consequently a rule ordering strategy was proposed, based on the "best first" case satisfaction approach, which can be applied when classifying "unseen" data. The concept of selectors [19] was summarised in section 3 together with some discussion of the randomised selectors. A novel rule mining approach was presented in section 4 based on the concepts of selectors (both determinstic and randomised). In theory, the proposed rule mining approach identifies significant CARs in time $O(k^2n^2)$ in its deterministic fashion, and $O(kn)$ in its randomised fashion. This mining approach avoids finding significant CARs on a "one-by-one" basis, which will require an exponential time $O(2^n)$. In section 5, two sets of evaluations were presented that evidence:

1. The proposed rule weighting and rule ordering approach's perform well with respect to the accuracy of classification; and
2. The proposed randomised rule mining approach is comparable to the "one-by-one" rule mining approach in significant CAR identification with respect to both the accuracy of classification and the efficiency of computation.

From the experimental results, it can be seen that the accuracy of classification obtained by the proposed randomised rule mining approach can be better than the accuracy obtained by the "one-by-one" approach. Further research is suggested to identify improved rule weighting scheme to find more significant rules in $\mathcal{R}$. Other obvious dirction for further research include: finding other rule ordering mechanisms that give a better classification accuracy; investigating other techniques to replace the proposed deterministic and/or randomised selectors to give a better performance; etc.

**Acknowledgement:** The authors would like to thank Professor Leszek Gąsieniec and Professor Paul Leng of the Department of Computer Science at the University of Liverpool for their support with respect to the work described here.

## References

1. Agrawal R, Imielinski T, Swami A (1993) Mining association rules between sets of items in large databases. In: Buneman P, Jajodia S (eds): Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data (SIGMOD-93, ACM Press, New York, NY), Washington, DC, United States, May 1993. (pages 207-216)
2. Agrawal R, Srikant R (1994) Fast algorithm for mining association rules. In: Bocca JB, Jarke M, Zaniolo C (eds): Proceedings of the 20th International Conference on Very Large Data Bases (VLDB-94, Morgan Kaufmann Publishers, San Francisco, CA), Santiago de Chile, Chile, September 1994. (ISBN 1-55860-153-8, pages 487-499)
3. Ali K, Manganaris S, Srikant R (1997) Partial classification using association rules. In: Heckerman D, Mannila H, Pregibon D, Uthurusamy R (eds): Proceedings of the Third International Conference on Knowledge Discovery and Data mining (KDD-97, AAAI Press, Menlo Park, CA), Newport Beach, California, United States, August 1997. (ISBN 1-57735-027-8, pages 115-118)
4. Blake CL, Merz CJ (1998) UCI repository of machine learning databases. http://www.ics.uci.edu/ mlearn/MLRepository.html, Irvine, CA: University of California, Department of Information and Computer Science.
5. Bong CH, Narayanan K (2004) An empirical study of feature selection for text categorization based on term weightage. In: Proceedings of the 2004 IEEE/WIC/ACM International Conference on Web Intelligence (WI-04, IEEE Computer Society), Beijing, China, September 2004. (ISBN 0-7695-2100-2, pages 599-602)

6. Boser BE, Guyon IM, Vapnik VN (1992) A training algorithm for optimal margin classifiers. In: Haussler D (eds): Proceedings of the 5th ACM annual Workshop on Computational Learning Theory (COLT-92, ACM Press, New York, NY), Pittsburgh, Pennsylvania, United States, July 1992. (ISBN 0-89791-497-X, pages 144-152)
7. Brin S, Motwani R, Ullman JD, Tsur S (1997) Dynamic itemset counting and implication rules for market basket data. In: Peckham J (eds): Proceedings of the 1997 ACM SIGMOD International Conference on Management of Data (SIGMOD-97, ACM Press, New York, NY), Tucson, Arizona, United States, May 1997. (SIGMOD Record 26(2), pages 255-264)
8. Burdick D, Calimlim M, Gehrke J (2001) MAFIA: A maximal frequent itemset algorithm for transactional databases. In: Proceedings of the 17th International Conference on Data Engineering (ICDE-01, IEEE Computer Society), Heidelberg, Germany, April 2001. (ISBN 0-7695-1001-9, pages 443-452)
9. Cheung W, Zaïane OR (2003) Incremental mining of frequent patterns without candidate generation or support constrain. In: 7th International Database Engineering and Applications Symposium (IDEAS-03, IEEE Computer Society), Hong Kong, China, July 2003. (ISBN 0-7695-1981-4, pages 111-116)
10. Clark P, Boswell R (1991) Rule induction with CN2: Some recent improvements. In: Kodratoff Y (eds): Machine Learning - Proceedings of the Fifth European Working Session on Learning (EWSL-91, Springer-Verlag Berlin), Porto, Portugal, March 1991. (LNAI 482, ISBN 3-540-53816-X, pages 151-163)
11. Coenen F, Goulbourne G, Leng P (2001) Computing association rules using partial totals. In: Raedt LD, Siebes A (eds): Principles of Data Mining and Knowledge Discovery – Proceedings of the 5th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD-01, Springer), Freiburg, Germany, September 2001. (LNAI 2168, ISBN 3-540-42534-9, pages 54-66)
12. Coenen F, Leng P (2001) Optimising association rule algorithms using itemset ordering. In: Bramer M, Coenen F, Preece A (eds): Research and Development in Intelligent Systems XVIII – Proceedings of the Twenty-first SGES International Conference on Knowledge Based Systems and Applied Artificial Intelligence (ES-01, Springer), Cambridge, United Kingdom, December 2001. (ISBN 1852335351, pages 53-66)
13. Coenen F, Leng P (2002) Finding association rules with some very frequent attributes. In: Elomaa T, Mannila H, Toivonen H (eds): Principles of Data Mining and Knowledge Discovery – Proceedings of the 6th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD-02, Springer), Helsinki, Finland, August 2002. (LNAI 2431, ISBN 3-540-44037-2, pages 99-111)
14. 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/, Department of Computer Science, The University of Liverpool, UK.
15. Coenen F, Leng P (2004) An evaluation of approaches to classification rule selection. In: Proceedings of the 4th IEEE International Conference on Data Mining (ICDM-04, IEEE Computer Society), Brighton, UK, November 2004. (ISBN 0-7695-2142-8, pages 359-362)
16. Coenen F, Leng P, Ahmed S (2004) Data structure for association rule mining: T-trees and p-trees. IEEE Transactions on Knowledge and Data Engineering, Volume 16(6):774-778

17. Coenen F, Leng P, Goulbourne G (2004) Tree structures for mining association rules. Journal of Data Mining and Knowledge Discovery, Volum 8(1):25-51
18. Coenen F, Leng P, Zhang L (2005) Threshold tuning for improved classification association rule mining. In: Ho TB, Cheung D, Liu H (eds): Advances in Knowledge Discovery and Data Mining – Proceedings of the Ninth Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD-05, Springer-Verlag Berlin Heidelberg), Hanoi, Vietnam, May 2005. (LNAI 3518, ISBN 3-540-26076-5, pages 216-225)
19. De Bonis A, Gąsieniec L, Vaccaro U (2003) Generalized framework for selectors with applications in optimal group testing. In: Baeten JCM, Lenstra JK, Parrow J, Woeginger GJ (eds): Proceedings of the 30th International Colloquium on Automata, Languages and Programming (ICALP-03, Springer), Eindhoven, The Netherlands, June 30 - July 4, 2003. (LNAI 2719, ISBN 3-540-40493-7, pages 81-96)
20. Domingos P, Pazzani M (1997) On the optimality of the simple bayesian classifier under zero-one loss. Machine Learning, 29(2/3): 103-130.
21. Dong G, Li J (1999) Efficient mining of emerging patterns: Discovering trends and differences. In: Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-99, ACM Press, New York, NY), San Diego, CA, United States, August 1999. (pages 43-52)
22. Dong G, Zhang X, Wong L, Li J (1999) CAEP: Classification by aggregating emerging patterns. In: Arikawa S, Furukawa K (eds): Discovery Science – Proceedings of the Second International Conference Discovery Science (DS-99, Springer), Tokyo, Japan, December 1999. (LNAI 1721, ISBN 3-540-66713-X, pages 30-42)
23. Dunham MH (2002) Data mining: Introductory and advanced topics. Prentice Hall, August 2002. (ISBN 0-13-088892-3)
24. El-Hajj M, Zaïane OR (2003) Inverted matrix: efficient discovery of frequent items in large datasets in the context of interactive mining. In: Getoor L, Senator TE, Domingos P, Faloutsos C (eds): Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-03, ACM Press, New York, NY), Washington, DC, United States, August 2003. (ISBN 1-58113-737-0, pages 109-118)
25. Freitas AA (2002) Data mining and knowledge discovery with evolutionary algorithms, Springer-Verlag Berlin Heidelberg New York, Germany, 2002. (ISBN 3-540-43331-7)
26. Gouda K, Zaki MJ (2001) Efficiently mining maximal frequent itemsets. In: Cercone N, Lin TY, Wu X (eds): Proceedings of the 2001 IEEE International Conference on Data Mining (ICDM-01, IEEE Computer Society), San Jose, CA, United Stated, 29 November - 2 December 2001. (ISBN 0-7695-1119-8, pages 163-170)
27. Han J, Pei J, Yin Y (2000) Mining frequent patterns without candidate generation. In: Chen W, Naughton JF, Bernstein PA (eds): Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data (SIGMOD-00, ACM Press, New York, NY), Dallas, TX, United States, May 2000. (ISBN 1-58113-218-2, pages 1-12)
28. Han J, Kamber M (2001) Data mining: Concepts and techniques. Morgan Kaufmann Publishers, San Francisco, CA, United States, 2001. (ISBN 1-55860-489-8)

29. Han J, Kamber M (2006) Data mining: Concepts and techniques (Second Edition). Morgan Kaufmann Publishers, San Francisco, CA, United States, March 2006. (ISBN 1-55860-901-6)
30. Hand D, Mannila H, Smyth R (2001) Principles of data mining. MIT Press, Cambridge, MA, United States, August 2001. (ISBN 0-262-08290-X)
31. Hidber C (1999) Online association rule mining. In: Delis A, Faloutsos C, Ghandeharizadeh S (eds): Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data (SIGMOD-99, ACM Press, New York, NY), Philadelphia, Pennsylvania, United States, June 1999. (ISBN 1-58113-084-8, pages 145-156)
32. Holsheimer M, Kersten ML, Mannila H, Toivonen H (1995): A perspective on databases and data mining. In: Fayyad UM, Uthurusamy R (eds): Proceedings of the First International Conference on Knowledge Discovery and Data Mining (KDD-95, AAAI Press, Menlo Park, CA), Montreal, Canada, August 1995. (ISBN 0-929280-82-2, pages 150-155)
33. Houtsma M, Swami A (1995) Set-oriented mining of association rules in relational databases. In: Yu PS, Chen AL (eds): Proceedings of the Eleventh International Conference on Data Engineering (ICDE-95, IEEE Computer Society), Taipei, Taiwan, March 1995. (ISBN 0-8186-6910-1, pages 25-33)
34. James M (1985) Classification algorithms. Wiley-Interscience, New York, NY, United States, 1985. (ISBN 0-471-84799-2)
35. Lavrač N, Flach P, Zupan B (1999) Rule evaluation measures: A unifying view. In: Dzeroski S, Flach PA (eds): Proceedings of the 9th International Workshop on Inductive Logic Programming (ILP-99, Springer), Bled, Slovenia, June 1999. (LNAI 1634, ISBN 3-540-66109-3, pages 174-185)
36. Li W, Han J, Pei J (2001) CMAR: Accurate and efficient clasification based on multiple class-association rules. In: Cercone N, Lin TY, Wu X (eds): Proceedings of the 2001 IEEE International Conference on Data Mininig (ICDM-01, IEEE Computer Society), San Jose, CA, United States, 29 November - 2 December 2001. (ISBN 0-7695-1119-8, pages 369-376)
37. Lin D-I, Kedem ZM (1998) Pincer search: A new algorithm for discovering the maximum frequent set. In: Schek H-J, Saltor F, Ramos I, Alonso G (eds): Advances in Database Technology – Proceedings of the 6th International Conference on Extending Database Technology (EDBT-98, Springer), Valencia, Spain, March 1998. (LNAI 1377, ISBN 3-540-64264-1, pages 105-119)
38. Liu B, Hsu W, Ma Y (1998) Integrating classification and association rule mining. In: Agrawal R, Stolorz PE, Piatetsky-Shapiro G (eds): Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining (KDD-98, AAAI Press, Menlo Park, CA), New York City, New York, United States, August 1998. (ISBN 1-57735-070-7, pages 80-86)
39. Liu J, Pan Y, Wang K, Han J (2002) Mining frequent item sets by opportunistic projection. In: Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-02, ACM Press, New York, NY), Edmonton, Alberta, Canada, July 2002. (ISBN 1-58113-567-X, pages 229-238)
40. Mannila H, Toivonen H, Verkamo AI (1994) Efficient algorithms for discovering association rules. In: Fayyad UM, Uthurusamy R (eds): Knowledge Discovery in Databases: Papers from the 1994 AAAI Workshop (KDD-94, AAAI Press, Menlo Park, CA), Seattle, Washington, United States, July 1994. (Technical Report WS-94-03, ISBN 0-929280-73-3, pages 181-192)

41. Michalski RS (1980) Pattern recognition as rule-guided inductive inference. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1980. (pages 774-778)
42. Mirkin B, Mirkin BG (2005) Clustering for data mining: A data recovery approach. Chapman & Hall/CRC, April 2005. (ISBN 1584885343)
43. Park JS, Chen M-S, Yu PS (1995) An effective hash based algorithm for mining association rules. In: Carey MJ, Schneider DA (eds): Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data (SIGMOD-95, ACM Press, New York, NY), San Jose, CA, United States, May 1995. (SIGMOD Record 24(2), pages 175-186)
44. Pei J, Han J, Mao R (2000) CLOSET: An efficient algorithm for mining frequent closed itemsets. In: Gunopulos D, Rastogi R (eds): 2000 ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery (SIGMOD-DMKD-01), Dallas, TX, United Stated, May 2000. (pages 21-30)
45. Quinlan JR (1993) C4.5: Programs for machine learning. Morgan Kaufmann Publishers, San Francisco, CA, United States, 1993. (ISBN 1-55860-238-0)
46. Quinlan JR, Cameron-Jones RM (1993) FOIL: A midterm report. In: Brazdil R (eds): Machine Learning – Proceedings of the 1993 European Conference on Machine Learning (ECML-93, Springer), Vienna, Austria, April 1993. (LNAI 667, ISBN 3-540-56602-3, pages 3-20)
47. Roberto J, Bayardo Jr (1998) Efficiently mining long patterns from databases. In: Hass LM, Tiwary A (eds): Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data (SIGMOD-98, ACM Press, New York, NY), Seattle, Washington, United States, June 1998. (ISBN 0-89791-995-5, pages 85-93)
48. Rymon R (1992) Search through systematic set enumeration. In: Nebel B, Rich C, Swartout WR (eds): Proceedings of the 3rd International Conference on Principles of Knowledge Representation and Reasoning (KR-92, Morgan Kaufmann Publishers, San Francisco, CA), Cambridge, MA, United States, October 1992. (ISBN 1-55860-262-3, pages 539-550)
49. Savasere A, Omiecinski E, Navathe S (1995) An efficient algorithm for mining association rules in large databases. In: Proceedings of the 21st International Conference on Very Large Data Bases (VLDB-95, Morgan Kaufmann Publishers, San Francisco, CA), Zurich, Switzerland, September 1995. (ISBN 1-55860-379-4, pages 432-444)
50. Toivonen H (1996) Sampling large databases for association rules. In: Vijayaraman TM, Buchmann AP, Mohan C, Sarda NL (eds): Proceedings of the 22nd International Conference on Very Large Data Bases (VLDB-96, Morgan Kaufmann Publishers, San Francisco, CA), Mumbai(Bombay), India, September 1996. (ISBN 1-55860-382-4, pages 134-145)
51. Wang J, Han J, Pei J (2003) CLOSET+: Searching for the best strategies for mining frequent closed itemsets. In: In: Getoor L, Senator TE, Domingos P, Faloutsos C (eds): Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-03, ACM Press, New York, NY), Washington, DC, United States, August 2003. (ISBN 1-58113-737-0, pages 236-245)
52. Wang W, Yang J (2005) Mining sequential patterns from large data sets. Springer-Verlag New York Inc., April 2005. (ISBN 0-387-24246-5)
53. Yin X, Han J (2003) CPAR: Classification based on predictive association rules. In: Barbará D, Kamath C (eds): Proceedings of the Third SIAM International

Conference on Data Mining (SDM-03, SIAM, Philadelphia, PA), San Francisco, CA, United States, May 2003. (ISBN 0-89871-545-8, pages 331-335)

54. Zaki MJ, Parthasarathy S, Ogihara M, Li W (1997) New algorithms for fast discovery of association rules. In: Heckerman D, Mannila H, Pregibon D (eds): Proceedings of the Third International Conference on Knowledge Discovery and Data Mining (KDD-97, AAAI Press, Menlo Park, CA), Beach, CA, United States, August 1997. (ISBN 1-57735-027-8, pages 283-286)
55. Zaki MJ, Hsiao C-J (2002) CHARM: An efficient algorithm for closed itemset mining. In: Grossman RL, Han J, Kumar V, Mannila H, Motwani R (eds): Proceedings of the Second SIAM International Conference on Data Mining (SDM-02, SIAM, Philadelphia, PA), Arlington, VA, United States, April 2002. (ISBN 0-89871-517-2, Part IX No. 1)