Effective Elimination of Redundant Association Rules

James Cheng Yiping Ke Wilfred Ng

Department of Computer Science and Engineering
The Hong Kong University of Science and Technology
Clear Water Bay, Kowloon, Hong Kong, China
{csjames, keyiping, wilfred}@cse.ust.hk

Abstract. It is well-recognized that the main factor that hinders the applications of *Association Rules* (*ARs*) is the huge number of ARs returned by the mining process. In this paper, we propose an effective solution that presents concise mining results by eliminating the redundancy in the set of ARs. We adopt the concept of *δ-Tolerance* to define the set of *δ-Tolerance ARs* (*δ-TARs*), which is a concise representation for the set of ARs. δ-Tolerance is a relaxation on the closure defined on the support of frequent itemsets, thus allowing us to effectively prune the redundant ARs. We then devise a set of *inference rules*, with which we prove that the set of δ-TARs is a *non-redundant* representation of ARs. In addition, we prove that the set of ARs that is derived from the δ-TARs by the inference rules is *sound* and *complete*. We also develop a compact tree structure called the *δ-TAR tree*, which facilitates the efficient generation of the δ-TARs and derivation of other ARs. Experimental results verify the efficiency of using the δ-TAR tree to generate the δ-TARs and to query the ARs. The set of δ-TARs is also shown to be drastically smaller than the state-of-the-art concise representations of ARs.

Keywords: Association Rules, Redundancy Elimination, Concise Representation

1 Introduction

Association Rule (*AR*) Mining (Agrawal et al., 1993) is recognized as one of the fundamental data mining tasks and has a wide range of applications (Geurts et al., 2003; Yang et al., 2004; Fonseca et al., 2005; Thabtah and Cowling, 2007; Kumar et al., 2007). Over the last decade, many efficient algorithms have been proposed for mining ARs (see (Ceglar and Roddick, 2006) for a comprehensive survey on AR mining). However, an intrinsic problem of mining ARs is that a prohibitively large number of rules can be easily generated, even in the case of a relatively high *minimum support threshold*, σ. The massive size of the mining result makes further analysis very difficult, thus severely restricting the practical usage of ARs. For example, on mining the dataset `pumsb*` (FIMI Dataset Repository, 2003) at $\sigma = 30\%$ and $\varsigma = 80\%$, where σ and ς are the *minimum support and confidence thresholds* respectively, we obtain 213,274,268 ARs. When we slightly lower σ to 20%, the problem already becomes intractable. Based on the number of frequent itemsets, we estimate that the space required to hold the ARs will be more than 140TB.

Previous work (Zaki, 2004; Li and Hamilton, 2004; Goethals et al., 2005) has shown that the majority of the ARs are redundant and proposed to remove the redundant ARs. However, the set of ARs returned by these proposals is still large and inefficient for more advanced analysis.

In our prior work (Cheng et al., 2006), we propose *δ-Tolerance Frequent Itemsets* (*δ-TCFIs*), which is a concise representation for the set of *Frequent Itemsets* (*FIs*). In this paper, we adopt the concept of δ-TCFI to define a concise representation for the set of ARs, called *δ-Tolerance Association Rules* (*δ-TARs*). We illustrate the concept of δ-TAR by the following example.

Example 1. Figure 1 shows 15 FIs (represented as nodes) that are generated from a database excerpt, where `abcd` is the abbreviation of the itemset {`a`, `b`, `c`, `d`} and the number following ":" is the frequency of `abcd`.

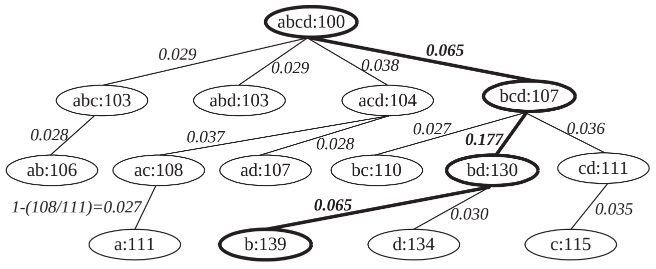


Fig. 1. Frequent Itemsets and Their Frequency

The number on each edge is computed as $d = (1 - \frac{frequency\ of\ Y}{frequency\ of\ X})$, where Y is X's smallest proper superset that has the greatest frequency. The concept of δ-TCFI is to prune X if $d \leq \delta$. For example, if we set $\delta = 0.04$, then the set of δ-TCFIs is {`abcd`, `bcd`, `bd`, `b`}, i.e., the bold nodes in Figure 1.

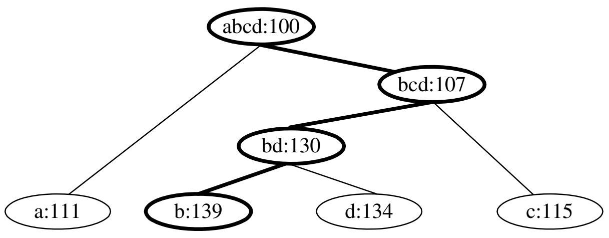


Fig. 2. A Condensed Tree of the Tree in Figure 1

The 15 FIs generate 50 ARs of confidence no less than 75%; however, the majority of these ARs carries similar information and are hence redundant. By adopting the concept of δ-TCFI, we can convert the tree in Figure 1 into a much smaller tree as shown in Figure 2. From this condensed tree, we obtain a very concise set of ARs, i.e., the set of δ-TARs, which consists of only 6 ARs, as defined by the 6 edges in Figure 2. The rest of the 44 ARs can be derived from the δ-TARs if the user demands. ■

In many applications (Zaki, 2004; Li and Hamilton, 2004; Goethals et al., 2005), the set of all ARs is too large to be handled by the user. Thus, it is more beneficial to give a small and concise set of ARs and allow the user to derive extra information when needed. For this purpose, we propose a set of *inference rules* to allow other ARs to be derived from the δ-TARs on demand. With the inference rules, we also prove that the set of δ-TARs is *non-redundant*, which means that no δ-TAR can be derived from other δ-TARs by applying the inference rules. More importantly, we show that the set of ARs derived from the δ-TARs is *sound* and *complete* with respect to the set of all ARs (at a given minimum support threshold), that is, the two sets of ARs are *equivalent*.

To efficiently generate the set of δ-TARs, we propose the *δ-TAR tree*, the set of edges of which defines the set of δ-TARs. The δ-TAR tree also serves as the supporting data structure for the efficient derivation of ARs by the inference rules. In addition, we can obtain the set of δ-TCFIs as well as recover the set of all FIs by traversing the δ-TAR tree only once. An example of a δ-TAR tree is shown in Figure 2, from which we can not only obtain the set of δ-TARs and derive all other ARs efficiently by following the paths, but can also obtain the set of δ-TCFIs (which is the bold nodes) and recover all FIs.

We devise an efficient algorithm to construct the δ-TAR tree from the set of δ-TCFIs. In addition, we design an *inverted index* to facilitate the querying of the δ-TAR tree. With the inverted index, we can efficiently recover the FIs and ARs, with their support and confidence, relevant to some δ-TCFIs and δ-TARs that are of interest to the user.

We compare our algorithm of mining δ-TARs with the algorithms for mining the following state-of-the-art concise representations of ARs: *Closed ARs* (Zaki, 2004), *Basic ARs* (Li and Hamilton, 2004), and *Non-Derivable ARs* (Goethals et al., 2005). While the algorithms are all efficient (there are no clear winners), the set of δ-TARs is one to two orders of magnitude smaller than the other three sets of rules (and is five to six orders of magnitude smaller than the set of all ARs). We also demonstrate that using the δ-TAR tree to derive the ARs and to query the δ-TCFIs/FIs returns the results instantly.

Organization. The rest of the paper is organized as follows. We give the related work and some preliminaries of mining FIs and ARs in Sections 2 and 3. Section 4 defines the notion of δ-TCFI. Section 5 defines the δ-TAR tree and the δ-TARs. Section 6 presents the algorithms for building and querying the δ-TAR tree. Then, Section 7 reports the experimental results and Section 8 concludes the paper.

2 Related Work

There are a number of proposals that address the redundancy problem of ARs. Kryszkiewicz (1998) and Bastide et al. (2000) define non-redundant ARs to be those that have minimal antecedents and maximal consequents. However, this type of ARs may not be useful to applications that aim at finding all possible causes of some phenomenons. Moreover, the number of such ARs can still be very large. Aggarwal and Yu (2001) defines a set of ARs to be redundant if they can be derived from an AR that has lower support and confidence. However, the support and confidence of the derived ARs cannot be recovered.

Zaki (2004) defines a set of non-redundant ARs, called *Closed ARs*, based on the set of closed FIs. The number of Closed ARs is linear to the number of CFIs, which can be large for sparse and large datasets. Li and Hamilton (2004) define an AR, $X \Rightarrow Y$, to be a *Basic AR* if $\nexists X' \subset X$, such that, $\forall X''$, where $X' \subseteq X'' \subseteq X$, $conf(X'' \Rightarrow Y) = conf(X \Rightarrow Y)$. Basic ARs are restricted to have only one item in the consequent. Recently, Goethals et al. (2005) proposes *non-derivable ARs* using the same inclusion-exclusion principle of the non-derivable FIs (Calders and Goethals, 2002). The confidence of a derivable AR is obtained from the confidence and support of all its proper subrules. As shown in our experiments, the three sets of rules mentioned above are still much larger than the set of δ-TARs.

Pasquier et al. (2005) also discusses the generation of non-redundant ARs using the concept of closed FIs (Pasquier et al., 1999). In addition, they propose non-redundant ARs having minimal antecedent and maximal consequent. We are also aware of the recent work by Palshikar et al. (2007), which reduces the number of ARs by the concept of *heavy itemsets*, where an itemset is *heavy* if all possible ARs (for given support and confidence values) made up of the items only in the itemset are present.

The concept of δ-TCFI is proposed in our previous work (Cheng et al., 2006). This concept has been successfully applied to query graph databases (Cheng et al., 2007a) and other possible applications include redundancy elimination in mining time series data (Mörchen and Ultsch, 2007) and data stream mining (Cheng et al., 2007b). This paper applies the concept to effectively remove the redundancy in the set of ARs. In addition, the δ-TAR tree proposed can also be used to efficiently query the δ-TCFIs and recover all or part of the FIs on the demand of the user, which are not discussed in (Cheng et al., 2006).

Using concise representations for interactive AR mining is also proposed in (Jeudy and Boulicaut, 2002), which applies free-sets (Boulicaut et al., 2003) to achieve a much smaller cache for optimizing AR queries in an interactive process. While caching can also be applied using δ-TCFIs, this paper proposes the δ-TAR tree to support efficient interactive AR querying. It is also possible to further reduce the number of rules by pruning *non-actionable rules* as proposed by Liu et al. (2001).

3 Preliminaries

Let $\mathcal{I} = \{x_1, x_2, \ldots, x_N\}$ be a set of items. An *itemset* (also called a *pattern*) is a non-empty subset of $\mathcal{I}$. A *transaction* is an itemset. We say that a transaction Y *supports* an itemset X if $Y \supseteq X$. For brevity, an itemset $\{x_{k_1}, x_{k_2}, \ldots, x_{k_m}\}$ is written as $x_{k_1}x_{k_2}\ldots x_{k_m}$ in this paper.

Let $\mathcal{D}$ be a database of transactions. The *frequency* of an itemset X, denoted as *freq*(X), is the number of transactions in $\mathcal{D}$ that support X. The *support* of X, denoted as *supp*(X), is defined as $\frac{freq(X)}{|\mathcal{D}|}$, where $|\mathcal{D}|$ is the number of transactions in $\mathcal{D}$. X is a *Frequent Itemset* (*FI*) if *supp*$(X) \geq \sigma$, where σ $(0 \leq \sigma \leq 1)$ is a user-specified *minimum support threshold*.

Let $\mathcal{F}$ be the set of all FIs. An itemset X is a *Maximal Frequent Itemset* (*MFI*) (Bayardo, 1998) if $X \in \mathcal{F}$ and $\nexists Y \in \mathcal{F}$ such that $Y \supset X$. X is a *Closed Frequent Itemset* (*CFI*) (Pasquier et al., 1999) if $X \in \mathcal{F}$ and $\nexists Y \in \mathcal{F}$ such that $Y \supset X$ and $supp(Y) = supp(X)$.

An *Association Rule* (*AR*) is defined as an implication of the form $X \Rightarrow (Y - X)$, where X and Y are itemsets, $X \neq \emptyset$ and $X \subset Y$. Let $r = (X \Rightarrow (Y - X))$. We call X the *antecedent* of r and $(Y - X)$ the *consequent* of r. The *support* of r, denoted as $supp(r)$, is defined as $supp(Y)$ and the *confidence* of r, denoted as $conf(r)$, is defined as $\frac{supp(Y)}{supp(X)}$. Given $\mathcal{F}$, it is straightforward to generate the set of ARs that have support no less than σ. Similar to the use of σ, a user-specified *minimum confidence threshold*, ς, where $0 \leq \varsigma \leq 1$, is also used to generate ARs with confidence no less than ς.

4 δ-Tolerance Closed Frequent Itemsets

In this section, we first define the notion of *δ-Tolerance Closed Frequent Itemsets* (*δ-TCFIs*), which is essential for the definition of *δ-TAR* in Section 5.

Definition 1 (δ-TOLERANCE CLOSED FREQUENT ITEMSET) *An itemset X is a* δ-Tolerance Closed Frequent Itemset (δ-TCFI) *if and only if $X \in \mathcal{F}$ and $\nexists Y \in \mathcal{F}$ such that $Y \supset X$ and $supp(Y) \geq ((1 - \delta) \cdot supp(X))$, where δ $(0 \leq \delta \leq 1)$ is a user-specified* support tolerance factor. *Let $\mathcal{T}(\delta)$, or simply $\mathcal{T}$ when δ is clear in the context, be the set of δ-TCFIs.*

We now define a few terms which help to define the concept of δ-TAR.

Definition 2 (CLOSEST SUPERSET) *Given $X \in (\mathcal{F} - \mathcal{T})$, let $\mathcal{Y}_X = \{Y : Y \supset X, |Y| = |X| + 1,$ and $supp(Y) \geq ((1 - \delta) \cdot supp(X))\}$. $Y \in \mathcal{Y}_X$ is the* closest superset *of X if $supp(Y) = MAX\{supp(Y') : Y' \in \mathcal{Y}_X\}$ and $\forall Y' \in \mathcal{Y}_X$, where $Y' \neq Y$ and $supp(Y') = supp(Y)$, Y is lexicographically ordered before Y'.*

Given $X \in (\mathcal{F} - \mathcal{T})$, we can follow a path of closest supersets and finally reach the closest superset that is a δ-TCFI. We define this δ-TCFI superset as the closest δ-TCFI superset of X as follows.

Definition 3 (CLOSEST δ-TCFI SUPERSET) *Let $X_1, \ldots, X_n \in \mathcal{F}$, where for $1 \leq i < n$, $X_i \subset X_{i+1}$ and $|X_{i+1}| = |X_i| + 1$. X_n is the* closest δ-TCFI superset *of X_1, if $X_n \in \mathcal{T}$ and, for $1 \leq i < n$, $X_i \in (\mathcal{F} - \mathcal{T})$ and X_{i+1} is the closest superset of X_i.*

With Definition 3, we further define the closure of a δ-TCFI as follows.

Definition 4 (Closure of δ-TCFI) *Given $Y \in \mathcal{T}$, the* closure *of Y, denoted as $CLOS(Y)$, is defined as $CLOS(Y)$=$\{X : Y$ is the closest δ-TCFI superset of $X\}$. We also define $clos(Y, i) = \{X : X \in CLOS(Y)$ and $|X| = |Y| - i\}$.*

The following example illustrates the concept of the closure of a δ-TCFI.

Example 2. Referring to Figure 1, when $\delta = 0.04$, the closest superset of `a` is `ac`, that of `ac` is `acd` and that of `acd` is `abcd`. Thus, `a`, `ac` and `acd` are recursively bounded by δ and included in the closure of `abcd`. For the two supersets of `ab` that have the same frequency, `abc` is the closest superset of `ab` since `abc` is ordered before `abd`.

The FI `abcd` is the closest δ-TCFI superset of all FIs in $S = \{$`a`, `ab`, `ac`, `ad`, `abc`, `abd`, `acd`$\}$; in other words, $CLOS(\texttt{abcd}) = S$, $clos(\texttt{abcd}, 3) = \{\texttt{a}\}$, $clos(\texttt{abcd}, 2) = \{\texttt{ab}, \texttt{ac}, \texttt{ad}\}$, and $clos(\texttt{abcd}, 1) = \{\texttt{abc}, \texttt{abd}, \texttt{acd}\}$. Similarly, we have $CLOS(\texttt{bcd}) = \{\texttt{c}, \texttt{bc}, \texttt{cd}\}$. ■

Given $Y \in \mathcal{T}$, we define the support extension of Y in order to give a clearer view of the closeness, in terms of support, between Y and the FIs in $CLOS(Y)$.

Definition 5 (Support Extension) *Given $Y \in \mathcal{T}$, the* support extension *of Y, denoted as $EXT(Y)$, is defined as a list $(ext(Y, 1), \ldots, ext(Y, m))$, where $m = MAX\{i : clos(Y, i) \neq \emptyset\}$ and $ext(Y, i)$, for $1 \leq i \leq m$, is given by*

$$ext(Y, i) = \frac{\sum_{X \in clos(Y,i)} (\frac{supp(X)}{supp(Y)})}{|clos(Y, i)|}.$$

The size of $EXT(Y)$ is given by $|EXT(Y)| = m$.

The support extension provides an accurate mechanism for the estimation of the support of the FIs recovered from a δ-TCFI. We define the support estimation of an FI as follows.

Definition 6 (Estimated Support) *Given $Y \in \mathcal{T}$ and $X \in CLOS(Y)$, the* estimated support *of X is defined as $(supp(Y) \cdot ext(Y, |Y| - |X|))$.*

Example 3. Referring to Example 2, we have $ext(\texttt{abcd}, 1) = (\frac{103}{100} + \frac{103}{100} + \frac{104}{100})/3 = 1.03$, $ext(\texttt{abcd}, 2) = (\frac{106}{100} + \frac{108}{100} + \frac{107}{100})/3 = 1.07$, and $ext(\texttt{abcd}, 3) = \frac{111}{100}/1 = 1.11$.

Thus, the support of `abc`, `abd` and `acd` are estimated as $(supp(\texttt{abcd}) \cdot ext(\texttt{abcd}, 1)) = (1.03 \times supp(\texttt{abcd}))$, and the support of `ab`, `ac` and `ad` are estimated as $(supp(\texttt{abcd}) \cdot ext(\texttt{abcd}, 2)) = (1.07 \times supp(\texttt{abcd}))$. ■

5 δ-Tolerance Association Rules

In this section, we present the *δ-Tolerance Association Rules* (*δ-TARs*). First, in Section 5.1 we present a set of ARs, $\mathcal{R}_{T_\mathcal{F}}$, which is generated from an *FI tree*, $T_\mathcal{F}$. We define three *inference rules* and apply them to show that the set of ARs derived from $\mathcal{R}_{T_\mathcal{F}}$ is *sound* and *complete*. Next, in Section 5.2 we present the *δ-TAR tree*, T_δ, which generates the set of *δ-TARs*, $\mathcal{R}_\delta$. We then define the fourth *inference rule* which is specifically for the derivation of ARs from $\mathcal{R}_\delta$. Then, in Section 5.3 we employ the result of Section 5.1 to prove that $\mathcal{R}_\delta$ is *non-redundant*, and the set of ARs derived from $\mathcal{R}_\delta$ is *sound* and *complete*. Finally, in Section 5.4 we analyze the error bound of the support and confidence of the derived ARs.

5.1 FI Tree and FI-Tree-Based Association Rules

We first define the term *FI-parent*, which will be used in the definition of the FI tree. Given $X \in \mathcal{F}$, let $\mathcal{Y}_X = \{Y : Y \in \mathcal{F},\ Y \supset X,$ and $\nexists Y' \in \mathcal{F}$ such that $Y' \supset X$ and $|Y'| < |Y|\}$. $Y \in \mathcal{Y}_X$ is the *FI-parent* of X if $supp(Y) = MAX\{supp(Y') : Y' \in \mathcal{Y}_X\}$ and $\forall Y' \in \mathcal{Y}_X$, where $Y' \neq Y$ and $supp(Y') = supp(Y)$, Y is lexicographically ordered before Y'.

We define the FI tree as follows.

Definition 7 (FI Tree) *Given $\mathcal{F}$, we define an* FI tree*, $T_\mathcal{F} = (V_\mathcal{F}, E_\mathcal{F})$, as follows:*

- *$V_\mathcal{F}$ is the set of vertices, defined as $V_\mathcal{F} = \mathcal{F} \cup \{\gamma\}$, where the root $\gamma = \{x : x \in \mathcal{I}$ and $supp(x) \geq \sigma\}$.*
- *$E_\mathcal{F}$ is the set of edges, defined as $E_\mathcal{F} = \{(Y, X) : X, Y \in \mathcal{F}$ and Y is the FI-parent of $X\} \cup \{(\gamma, X) : X \subset \gamma$ and $\nexists Y \in \mathcal{F}$ such that $Y \supset X\}$.*

The root γ, which represents the set of frequent items, is needed to connect the disconnected components in $T_\mathcal{F}$, if the set of frequent items is not an FI. In the following discussion, we also show that γ is needed to ensure the completeness of the AR derivation.

Example 4. If $\mathcal{F}$ is the set of 15 FIs shown in Figure 1, then $T_{\mathcal{F}}$ is the tree in Figure 1, where $\gamma = \texttt{abcd}$. ■

From $T_{\mathcal{F}}$, we generate the following set of ARs, denoted as $\mathcal{R}_{T_{\mathcal{F}}}$.

Definition 8 (FI-Tree-Based Association Rules) *Given an FI tree $T_{\mathcal{F}}$, the set of* FI-Tree-Based ARs*, $\mathcal{R}_{T_{\mathcal{F}}}$, is defined as $\mathcal{R}_{T_{\mathcal{F}}} = \{X \Rightarrow (Y - X) : (Y, X) \in E_{\mathcal{F}}\}$, where the support and confidence of each $r = (X \Rightarrow (Y - X)) \in \mathcal{R}_{T_{\mathcal{F}}}$ are defined as follows:*

$$supp(r) = supp(Y) \ ,$$

$$conf(r) = \frac{supp(Y)}{supp(X)} \ .$$

We set $supp(Y) = \kappa$ when $Y = \gamma$ and $\gamma \notin \mathcal{F}$, where κ is a non-zero number which is used to preserve the value of $supp(X)$ in $\frac{supp(Y)}{supp(X)}$ for the derivation of other ARs (by Rule 2 defined below).

Let $\mathcal{R}$ be the set of all ARs that have support no less than σ. When $\gamma \in \mathcal{F}$, then $\mathcal{R}_{T_{\mathcal{F}}} \subseteq \mathcal{R}$. Let $\mathcal{R}_\gamma = \{X \Rightarrow Y : (X \cup Y) = \gamma$ and $\gamma \notin \mathcal{F}\}$. When $\gamma \notin \mathcal{F}$, then $(\mathcal{R}_{T_{\mathcal{F}}} - \mathcal{R}_\gamma) \subseteq \mathcal{R}$. The support and confidence of the ARs in $\mathcal{R}_\gamma$ are not *correct* since $supp(\gamma)$ is defined as a special value κ. Thus, $\mathcal{R}_\gamma$ is regarded as a set of *auxiliary ARs*, which are only used as intermediate ARs to derive the ARs in $\mathcal{R}$.

In the following discussion, we first define three inference rules to derive $\mathcal{R}$ from $\mathcal{R}_{T_{\mathcal{F}}}$. Then, we prove in Theorem 1 that the set of ARs derived from $\mathcal{R}_{T_{\mathcal{F}}}$ excluding those ARs in $\mathcal{R}_\gamma$ is sound and complete with respect to $\mathcal{R}$.

Rule 1 (Transitivity Derivation) *Given two ARs $r_1 = (X \Rightarrow Y)$ and $r_2 = ((X \cup Y) \Rightarrow Z)$,* transitivity derivation *generates an AR, $r = (X \Rightarrow (Y \cup Z))$, where the support and confidence of r are defined as follows:*

$$supp(r) = supp(r_2) \ ,$$

$$conf(r) = (conf(r_1) \ \cdot \ conf(r_2)) \ .$$

Lemma 1. *Referring to Rule 1, let $r_1, r_2 \in (\mathcal{R}_{T_{\mathcal{F}}} \cup \mathcal{R})$. Then, $supp(r)$ and $conf(r)$ are correctly defined, and $r \in \mathcal{R}$ or $r \in \mathcal{R}_\gamma$.*

Proof. First, $supp(r) = supp(X \cup Y \cup Z) = supp(r_2)$ and $conf(r) = \frac{supp(X \cup Y \cup Z)}{supp(X)} = (\frac{supp(X \cup Y)}{supp(X)} \cdot \frac{supp(X \cup Y \cup Z)}{supp(X \cup Y)}) = (conf(r_1) \cdot conf(r_2))$.

Next, if $r_2 \in \mathcal{R}$, then $supp(r) = supp(r_2) \geq \sigma$, which implies $r \in \mathcal{R}$. If $r_2 \in \mathcal{R}_{T_\mathcal{F}}$ and $r_2 \notin \mathcal{R}_\gamma$, then $supp(r) = supp(X \cup Y \cup Z) \geq \sigma$ and hence $r \in \mathcal{R}$. If $r_2 \in \mathcal{R}_{T_\mathcal{F}}$ and $r_2 \in \mathcal{R}_\gamma$, then $(X \cup Y \cup Z) = \gamma$ and hence $r \in \mathcal{R}_\gamma$.

Lemma 1 shows that r_1 and r_2 can be in either $\mathcal{R}$ or $\mathcal{R}_{T_\mathcal{F}}$. This is because the inference rules are first applied on ARs from $\mathcal{R}_{T_\mathcal{F}}$ and then recursively applied on the derived ARs to generate the complete set of ARs.

To apply transitivity derivation, we start with $r_1, r_2 \in \mathcal{R}_{T_\mathcal{F}}$. Since r_1 and r_2 are defined by two adjacent edges, $(X \cup Y, X)$ and $(X \cup Y \cup Z, X \cup Y)$, in $T_\mathcal{F}$, we can follow a path in $T_\mathcal{F}$ and apply transitivity derivation recursively to generate more ARs, as illustrated by the following example.

Example 5. Referring to Example 4, let $r_1 = (\mathtt{b} \Rightarrow \mathtt{d})$, $r_2 = (\mathtt{bd} \Rightarrow \mathtt{c})$ and $r_3 = (\mathtt{bcd} \Rightarrow \mathtt{a})$. We can derive $r_4 = (\mathtt{b} \Rightarrow \mathtt{cd})$ from r_1 and r_2, where $supp(r_4) = supp(r_2) = supp(\mathtt{bcd})$ and $conf(r_4) = (conf(r_1) \cdot conf(r_2)) = (\frac{supp(\mathtt{bd})}{supp(\mathtt{b})} \cdot \frac{supp(\mathtt{bcd})}{supp(\mathtt{bd})}) = \frac{supp(\mathtt{bcd})}{supp(\mathtt{b})}$. From r_4 and r_3, we can further derive $r_5 = (\mathtt{b} \Rightarrow \mathtt{acd})$, where $supp(r_5) = supp(r_3) = supp(\mathtt{abcd})$ and $conf(r_5) = (conf(r_4) \cdot conf(r_3)) = (\frac{supp(\mathtt{bcd})}{supp(\mathtt{b})} \cdot \frac{supp(\mathtt{abcd})}{supp(\mathtt{bcd})}) = \frac{supp(\mathtt{abcd})}{supp(\mathtt{b})}$. ■

Rule 2 (Pseudo-Transitivity Derivation) *Given two ARs $r_1 = (X \Rightarrow Y)$ and $r_2 = (Z \Rightarrow W)$, where $X \subset Z$ and $(X \cup Y) = (Z \cup W)$,* pseudo-transitivity derivation *generates an AR, $r = (X \Rightarrow (Z - X))$, where the support and confidence of r are defined as follows:*

$$supp(r) = \frac{supp(r_1)}{conf(r_2)} \ ,$$

$$conf(r) = \frac{conf(r_1)}{conf(r_2)} \ .$$

Lemma 2. *Referring to Rule 2, let $r_1, r_2 \in (\mathcal{R}_{T_\mathcal{F}} \cup \mathcal{R})$. Then, $supp(r)$ and $conf(r)$ are correctly defined, and $r \in \mathcal{R}$.*

Proof. First, $supp(r) = supp(Z) = (\frac{supp(Z)}{supp(Z \cup W)} \cdot supp(X \cup Y)) = (\frac{1}{conf(r_2)} \cdot supp(r_1))$ and $conf(r) = \frac{supp(Z)}{supp(X)} = (\frac{supp(Z)}{supp(Z \cup W)} \cdot \frac{supp(X \cup Y)}{supp(X)}) = (\frac{1}{conf(r_2)} \cdot$

$conf(r_1))$.

Next, since $r_2 \in (\mathcal{R}_{T_\mathcal{F}} \cup \mathcal{R})$, $Z \in \mathcal{F}$ and hence $supp(Z) \geq \sigma$, which implies that $supp(r) \geq \sigma$ and $r \in \mathcal{R}$.

The condition $(X \cup Y) = (Z \cup W)$ suggests that pseudo-transitivity derivation is not applied on ARs formed from edges on the same path. Instead, pseudo-transitivity allows us to derive an AR from edges on two different paths that share the same root. The condition $(X \cup Y) = (Z \cup W)$ is satisfied by applying transitivity derivation on the edges on each path up to the common root of the two paths. This also explains the need of the common root γ as to ensure all ARs can be generated.

Example 6. Consider the derivation of the AR $r = (\mathtt{b} \Rightarrow \mathtt{ac})$. We first apply transitivity derivation from b up to abcd to obtain $r_1 = (\mathtt{b} \Rightarrow \mathtt{acd})$ as done in Example 5. Then, we obtain r by applying pseudo-transitivity derivation on r_1 and r_2, where $r_2 = (\mathtt{abc} \Rightarrow \mathtt{d})$, and we have $supp(r) = \frac{supp(r_1)}{conf(r_2)} = \frac{supp(\mathtt{abcd})}{supp(\mathtt{abcd})/supp(\mathtt{abc})} = supp(\mathtt{abc})$ and $conf(r) = \frac{conf(r_1)}{conf(r_2)} = \frac{supp(\mathtt{abcd})/supp(\mathtt{b})}{supp(\mathtt{abcd})/supp(\mathtt{abc})} = \frac{supp(\mathtt{abc})}{supp(\mathtt{b})}$. ■

To ensure the completeness of AR derivation, we also need the following inference rule.

Rule 3 (Reflexivity Derivation) *Given an AR $r = (X \Rightarrow Y)$,* reflexivity derivation *generates r itself, where $supp(r)$ and $conf(r)$ remain unchanged.*

Lemma 3. *Referring to Rule 3, let $r \in (\mathcal{R}_{T_\mathcal{F}} \cup \mathcal{R})$. Then, $supp(r)$ and $conf(r)$ are correctly defined, and $r \in \mathcal{R}$ or $r \in \mathcal{R}_\gamma$.*

Proof. It is trivial that $supp(r)$ and $conf(r)$ are correctly defined. If $r \in \mathcal{R}_{T_\mathcal{F}}$ and $r \notin \mathcal{R}_\gamma$, then $supp(r) = supp(X \cup Y) \geq \sigma$ and hence $r \in \mathcal{R}$. For the other cases, $r \in \mathcal{R}$ or $r \in \mathcal{R}_\gamma$.

We now prove that the set of ARs derived from $\mathcal{R}_{T_\mathcal{F}}$ excluding those ARs in $\mathcal{R}_\gamma$ is sound and complete with respect to $\mathcal{R}$.

Theorem 1 *Given $\mathbb{R} = \{$Rule 1, Rule 2, Rule 3$\}$, let $\mathcal{R}^+_{T_\mathcal{F}}$ be the set of all ARs derived from $\mathcal{R}_{T_\mathcal{F}}$ by recursively applying $\mathbb{R}$. Let $\mathcal{R}' = (\mathcal{R}^+_{T_\mathcal{F}} - \mathcal{R}_\gamma)$. Then, $\mathcal{R}'$ is* sound *and* complete *with respect to $\mathcal{R}$, that is, $\mathcal{R}' = \mathcal{R}$.*

Proof. According to Lemmas 1 to 3, $\forall r \in \mathcal{R}^+_{T_\mathcal{F}}$, $r \in \mathcal{R}$ or $r \in \mathcal{R}_\gamma$, which implies that $\forall r \in (\mathcal{R}^+_{T_\mathcal{F}} - \mathcal{R}_\gamma)$, $r \in \mathcal{R}$. Thus, $\mathcal{R}'$ is sound with respect to $\mathcal{R}$, i.e., $\mathcal{R}' \subseteq \mathcal{R}$.

To prove the completeness, i.e., $\mathcal{R} \subseteq \mathcal{R}'$, let $r = (X \Rightarrow Y) \in \mathcal{R}$. Since X and $(X \cup Y)$ are in $\mathcal{F}$, it follows that X and $(X \cup Y)$ are also in $V_{\mathcal{F}}$. If $(X \cup Y)$ is an ancestor of X in $T_{\mathcal{F}}$, we can derive r by applying transitivity derivation on the path from X to $(X \cup Y)$. Otherwise, a common ancestor $(X \cup Y \cup Z)$ must exist. We first apply transitivity derivation on the two paths, $\langle X, \ldots, X \cup Y \cup Z \rangle$ and $\langle X \cup Y, \ldots, X \cup Y \cup Z \rangle$, to derive $r_1 = (X \Rightarrow (Y \cup Z))$ and $r_2 = ((X \cup Y) \Rightarrow Z)$. Then, we obtain r by applying pseudo-transitivity derivation on r_1 and r_2. If $r \in \mathcal{R}_{T_{\mathcal{F}}}$, then we obtain r by applying reflexivity derivation. Since $\forall r \in \mathcal{R}$, r can be derived from $\mathcal{R}_{T_{\mathcal{F}}}$ by applying $\mathbb{R}$, the result $\mathcal{R} \subseteq \mathcal{R}'$ thus follows.

5.2 δ-Tolerance Association Rule Tree and δ-Tolerance Association Rules

Although we can derive $\mathcal{R}$ from $\mathcal{R}_{T_{\mathcal{F}}}$, the size of $\mathcal{R}_{T_{\mathcal{F}}}$ is the same as the size of $\mathcal{F}$. We next show that the majority of the vertices in $V_{\mathcal{F}}$ that correspond to the FIs in $(\mathcal{F} - \mathcal{T})$ are redundant. Since $\mathcal{T}$ is orders of magnitude smaller than $\mathcal{F}$, we can significantly compress $T_{\mathcal{F}}$ if the redundant vertices and their corresponding edges are concisely represented. Thus, we define a new tree based on $\mathcal{T}$ instead of $\mathcal{F}$.

The new tree is called the *δ-Tolerance Association Rule tree* (*δ-TAR tree*), as from which we derive the set of *δ-Tolerance Association Rules*.

We first define the term *δ-TCFI-parent*, which will be used in the definition of the δ-TAR tree. Given $X \in \mathcal{T}$, let $\mathcal{Y}_X = \{Y : Y \in \mathcal{T},\ Y \supset X,$ and $\nexists Y' \in \mathcal{T}$ such that $Y' \supset X$ and $|Y'| < |Y|\}$. $Y \in \mathcal{Y}_X$ is the *δ-TCFI-parent* of X if $supp(Y) = MAX\{supp(Y') : Y' \in \mathcal{Y}_X\}$ and $\forall Y' \in \mathcal{Y}_X$, where $Y' \neq Y$ and $supp(Y') = supp(Y)$, Y is lexicographically ordered before Y'.

Definition 9 (δ-Tolerance Association Rule Tree) *The* δ-Tolerance Association Rule tree (δ-TAR tree)*, denoted as $T_\delta = (V_\delta, E_\delta)$, where $V_\delta = V_{\mathcal{T}} \cup \{\gamma\} \cup V_b$ and $E_\delta = E_{\mathcal{T}} \cup E_\gamma \cup E_b$, are defined as follows:*

– *$V_{\mathcal{T}} = \mathcal{T}$.*
  $E_{\mathcal{T}} = \{(Y, X) : X, Y \in \mathcal{T}$ and Y is the δ-TCFI-parent of $X\}$.
– *γ is the root of T_δ, where $\gamma = \{x : x \in \mathcal{I}$ and $supp(x) \geq \sigma\}$.*
  $E_\gamma = \{(\gamma, X) : X \subset \gamma$ and $\nexists Y \in \mathcal{T}$ such that $Y \supset X\}$.
– *$V_b = \bigcup_{Y \in \mathcal{T}} \{X :\ X \in CLOS(Y)$ and $\nexists X' \in CLOS(Y)$ such that $X' \subset X\}$.*
  $E_b = \{(Y, X) : X \in V_b,\ Y \in V_{\mathcal{T}},$ and $X \in CLOS(Y)\}$.

T_δ consists of both δ-TCFIs and FIs, i.e., $V_\mathcal{T}$ and V_b. For each edge $(Y, X) \in E_\mathcal{T}$, X's parent, Y, is X's smallest superset in $\mathcal{T}$ that has the greatest support and is lexicographically ordered before all other X's supersets in $\mathcal{T}$ that have the same support and size as Y.

The set of vertices V_b is called the set of *border* vertices, since given $Y \in \mathcal{T}$, all $X \in CLOS(Y)$ are contained between Y's border vertices and Y. Therefore, given X which is a border vertex of Y, then all X', where $X \subset X' \subset Y$, are redundant because they are FIs contained between X and Y and can be easily enumerated.

We note that in some rare cases, there may be some X' between X and Y that is not in $CLOS(Y)$. This causes a problem, since X' may be enumerated more than once and the support of the X' enumerated elsewhere may be different. In such a case, we may choose the greater support as the estimated support of X'.

Compared with $T_\mathcal{F}$, a huge number of redundant vertices in $T_\mathcal{F}$, as well as their incident edges, are eliminated from T_δ. Thus, T_δ is a significantly smaller tree than $T_\mathcal{F}$. The following example helps illustrate the idea of T_δ.

Example 7. Let $\delta = 0.04$, Figure 3 shows the δ-TAR tree, T_δ, of our running example (the root $\gamma = \texttt{abcd}$). Referring to Example 2, only $\texttt{a} \in CLOS(\texttt{abcd})$ is a border vertex. The FIs in $(CLOS(\texttt{abcd}) - \{a\})$, i.e., {ab, ac, ad, abc, abd, acd}, are redundant and can be represented concisely by the edge $(\texttt{abcd}, \texttt{a})$. We can also find another border vertex c in $CLOS(\texttt{bcd})$; thus, from the edge $(\texttt{bcd}, \texttt{c})$ we can enumerate the redundant FIs bc and cd. ■

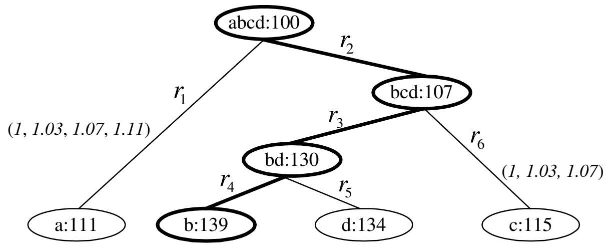


Fig. 3. The δ-TAR Tree T_δ

The δ-TAR tree is the supporting data structure for the efficient application of the inference rules, which allow users to derive more ARs on demand. We can also obtain all δ-TCFIs as well as recover all FIs with only one scan of T_δ. Most importantly, the δ-TARs are directly generated from T_δ.

Definition 10 (δ-Tolerance Association Rules) *Given T_δ, we define the set of* δ-Tolerance Association Rules (δ-TARs) *as $\mathcal{R}_\delta = \{X \Rightarrow (Y - X) : (Y, X) \in E_\delta\}$, where the support and confidence of each $r = (X \Rightarrow (Y - X)) \in \mathcal{R}_\delta$ are defined as follows:*

$$supp(r) = supp(Y) \ ,$$

$$conf(r) = \begin{cases} \frac{supp(Y)}{supp(X)} & \text{if } X \in V_{\mathcal{T}}, \\ \frac{1}{ext(Y, |Y| - |X|)} & \text{otherwise.} \end{cases}$$

The following example shows the set of δ-TARs of our running example.

Example 8. Given T_δ in Figure 3, we obtain $\mathcal{R}_\delta = \{r_1 = (\texttt{a} \Rightarrow \texttt{bcd}),\ r_2 = (\texttt{bcd} \Rightarrow \texttt{a}),\ r_3 = (\texttt{bd} \Rightarrow \texttt{c}),\ r_4 = (\texttt{b} \Rightarrow \texttt{d}),\ r_5 = (\texttt{d} \Rightarrow \texttt{b}),\ r_6 = (\texttt{c} \Rightarrow \texttt{bd})\}$. We have $supp(r_1) = supp(r_2) = supp(\texttt{abcd})$, and $conf(r_2) = \frac{supp(\texttt{abcd})}{supp(\texttt{bcd})} = \frac{100}{107}$. Since $\texttt{a} \notin V_{\mathcal{T}}$, $conf(r_1) = \frac{1}{ext(\texttt{abcd}, |\texttt{abcd}| - |\texttt{a}|)} = \frac{1}{1.11}$. ■

As shown in Definition 10 and Example 8, if $X \in (\mathcal{F} - \mathcal{T})$, we shall estimate the support (or frequency) of X from X's closest δ-TCFI superset Y. Thus, similar to the way we define the support extension of a δ-TCFI to estimate the support of an FI, we also define the *support extension* of a δ-TAR in order to compute the support and confidence of the ARs.

Definition 11 (Support Extension) *Given $r = (X \Rightarrow (Y - X)) \in \mathcal{R}_\delta$, where $X \in V_b$, the* support extension *of r, denoted as $EXT(r)$, is a list $(ext(r, 0), ext(r, 1), \ldots, ext(r, m))$, where $m = (|Y| - |X|)$ and $ext(r, i)$ is defined as follows:*

$$ext(r, i) = \begin{cases} 1 & \text{if } i = 0, \\ ext(Y, i) & \text{if } 1 \le i \le m. \end{cases}$$

We remark that rule analysis is often processed with the help of an efficient data structure such as T_δ; thus, the support extension is not explicitly required, since we can simply compute the support and confidence of an AR using the frequency extension of the δ-TCFIs in T_δ.

Given $\mathcal{R}_\delta$, we now can apply transitivity derivation and pseudo-transitivity derivation to derive more ARs when demanded. However, a large number of redundant edges in $T_\mathcal{F}$ are eliminated and not present in T_δ. Thus, we need one more inference rule to derive the ARs that are generated from these eliminated edges in $T_\mathcal{F}$.

Rule 4 (Closure Derivation) *Given* $r_1 = (X \Rightarrow (Y - X)) \in \mathcal{R}_\delta$, *where* $X \in V_b$, closure derivation *generates an AR,* $r = (Z \Rightarrow (W - Z))$, *where* $X \subseteq Z \subset W \subseteq Y$. *The support and confidence of* r *are defined as follows:*

$$supp(r) = supp(r_1) \;\cdot\; ext(r_1, |Y| - |W|) \;,$$

$$conf(r) = \frac{ext(r_1, |Y| - |W|)}{ext(r_1, |Y| - |Z|)} \;.$$

Lemma 4. *Referring to Rule 4, let* $r_1 \in \mathcal{R}_\delta \cup \mathcal{R}$. *Then,* $supp(r)$ *and* $conf(r)$ *are correctly defined, and* $r \in \mathcal{R}$.

Proof. First, $supp(r) = supp(W) = (supp(Y) \;\cdot\; ext(Y, |Y| - |W|)) = (supp(r_1) \cdot ext(r_1, |Y| - |W|))$ and $conf(r) = \frac{supp(W)}{supp(Z)} = \frac{supp(Y) \,\cdot\, ext(Y,|Y|-|W|)}{supp(Y) \,\cdot\, ext(Y,|Y|-|Z|)} = \frac{ext(r_1,|Y|-|W|)}{ext(r_1,|Y|-|Z|)}$.

Next, since $supp(r_1) = supp(Y) \geq \sigma$ and $ext(Y, |Y| - |W|)) \geq 1$, $supp(r) \geq \sigma$ and hence $r \in \mathcal{R}$.

We illustrate closure derivation by the following example.

Example 9. In Figure 3, the numbers in the two brackets on the two edges, $(\mathtt{a}, \mathtt{abcd})$ and $(\mathtt{c}, \mathtt{bcd})$, are the support extensions of the two ARs r_1 and r_6 (defined in Example 8), respectively.

By closure derivation, we can derive 18 ARs of the form $(Z \Rightarrow (W - Z))$ from $r_1 = (\mathtt{a} \Rightarrow \mathtt{bcd})$, where $\mathtt{a} \subseteq Z \subset W \subseteq \mathtt{abcd}$. Consider two of these derived ARs, $r_{11} = (\mathtt{ab} \Rightarrow \mathtt{cd})$ and $r_{12} = (\mathtt{ab} \Rightarrow \mathtt{c})$.

For r_{11}, we estimate $supp(r_{11}) = (supp(r_1) \cdot ext(r_1, 0)) = (supp(r_1) \times 1) = supp(\mathtt{abcd})$ and $conf(r_{11}) = \frac{ext(r_1,0)}{ext(r_1,2)} = \frac{1}{1.07} \approx \frac{supp(\mathtt{abcd})}{supp(\mathtt{ab})} = \frac{100}{106}$.

For r_{12}, we estimate $supp(r_{12}) = (supp(r_1) \cdot ext(r_1, 1)) = (supp(\mathtt{abcd}) \times 1.03) = supp(\mathtt{abc})$ and $conf(r_{12}) = \frac{ext(r_1,1)}{ext(r_1,2)} = \frac{1.03}{1.07} \approx \frac{supp(\mathtt{abc})}{supp(\mathtt{ab})} = \frac{103}{106}$.

Likewise, we can derive 4 ARs from $r_6 = (\mathtt{c} \Rightarrow \mathtt{bd})$ by closure derivation. ■

5.3 Non-Redundancy, Soundness and Completeness

In this subsection, we prove that $\mathcal{R}_\delta$ is *non-redundant* and that the set of all ARs derived from $\mathcal{R}_\delta$ by Rules 1 to 4 excluding those ARs in $\mathcal{R}_\gamma$ is sound and complete with respect to $\mathcal{R}$. We first define *non-redundancy* as follows.

Definition 12 (Non-Redundancy) *Given a set of ARs, R, R is* non-redundant *if $\forall r \in R$, r cannot be derived by applying Rules 1 to 4 on any other ARs in R.*

Let $\mathcal{R}_\gamma = \{X \Rightarrow Y : (X \cup Y) = \gamma \text{ and } \gamma \notin \mathcal{F}\}$ with respect to γ in T_δ.

Theorem 2 *Given $\mathbb{R} = \{$Rule 1, Rule 2, Rule 3, Rule 4$\}$, let $\mathcal{R}_\delta^+$ be the set of all ARs derived from $\mathcal{R}_\delta$ by recursively applying $\mathbb{R}$. Let $\mathcal{R}' = (\mathcal{R}_\delta^+ - \mathcal{R}_\gamma)$. Then, $\mathcal{R}_\delta$ is* non-redundant, *and $\mathcal{R}'$ is* sound *and* complete *with respect to $\mathcal{R}$.*

Proof. We first prove that $\mathcal{R}_\delta$ is non-redundant.

Suppose to the contrary that $\exists r \in \mathcal{R}_\delta$ such that r can be derived from some other ARs in $\mathcal{R}_\delta$. Let $r = (X \Rightarrow Y)$. Since $r \in \mathcal{R}_\delta$, we have $X \in V_\delta$ and X is unique in V_δ. Thus, if we can derive r from other ARs in $\mathcal{R}_\delta$ without using r (and hence X in V_δ), then we must first generate an AR, r', with X as its antecedent by closure derivation. Then, we generate r by applying $\mathbb{R}$ on r' and other ARs in $\mathcal{R}_\delta$. However, $X \in V_\delta$ means that X is in the closure of a δ-TCFI only if X is a border vertex, which implies that no AR with X as its antecedent can be generated by closure derivation (unless from r itself). Thus, r cannot be generated from other ARs in $\mathcal{R}_\delta$ and this leads to a contradiction.

Now, we prove that $\mathcal{R}'$ is sound and complete with respect to $\mathcal{R}$.

First, according to Lemma 4, all ARs derived from $\mathcal{R}_\delta$ by closure derivation are in $\mathcal{R}$. Since $\forall r \in \mathcal{R}_\delta$, $r \in \mathcal{R}$ or $r \in \mathcal{R}_\gamma$, it follows from Lemmas 1 to 3 that $\mathcal{R}'$ is sound with respect to $\mathcal{R}$.

We prove the completeness by applying the result of Theorem 1. We first show that $\mathcal{R}_{T_\mathcal{F}}$ can be derived from $\mathcal{R}_\delta$. Let $r = (X \Rightarrow (Y - X)) \in \mathcal{R}_{T_\mathcal{F}}$. If $X, Y \notin V_\delta$, then X and Y must be in the closure of some δ-TCFIs, X' and Y', and we can apply closure derivation to derive the ARs $X \Rightarrow (X' - X)$ and $Y \Rightarrow (Y' - Y)$. Then, we can derive r by applying transitivity derivation and pseudo-transitivity derivation on the paths from X' and Y' to the common ancestor of $(X' \cup Y')$ in T_δ. In a similar way, we can derive r from $\mathcal{R}_\delta$ for the other three cases: only $X \notin V_\delta$, only $Y \notin V_\delta$, or $X, Y \in V_\delta$. In Theorem 1, we have shown that $\mathcal{R} \subseteq (\mathcal{R}^+_{T_\mathcal{F}} - \mathcal{R}_\gamma)$. Since we can obtain $\mathcal{R}_{T_\mathcal{F}}$ from $\mathcal{R}_\delta$, the result $\mathcal{R} \subseteq \mathcal{R}' = (\mathcal{R}^+_\delta - \mathcal{R}_\gamma)$ follows.

$\mathcal{R}_\delta$ is significantly smaller than $\mathcal{R}$ since the majority of the ARs in $\mathcal{R}$ are redundant. As shown in Example 8, there are only 6 ARs in $\mathcal{R}_\delta$ while there are 50 ARs in $\mathcal{R}$. We derive a bound on the size of $\mathcal{R}_\delta$ with respect to that of $\mathcal{R}$ as follows.

Theorem 3 *Let l be the size of the largest FI in $\mathcal{F}$. Then, $\frac{|\mathcal{R}|}{|\mathcal{R}_\delta|} \geq \mathcal{O}(2^l)$.*

Proof. In the worst case, $\mathcal{F} = \mathcal{T}$ and T_δ becomes as big as $\mathcal{T}_\mathcal{F}$, which implies that $|\mathcal{R}_\delta| = |\mathcal{F}|$. Since $|\mathcal{R}|$ is $\mathcal{O}(|\mathcal{F}| \cdot 2^l)$, we obtain the bound.

Theorem 3 gives the worst-case upper bound on the number of δ-TARs. However, this worst case is extremely rare. In the average case, $|\mathcal{R}_\delta| = \mathcal{O}(|\mathcal{T}|)$ and thus $\frac{|\mathcal{R}|}{|\mathcal{R}_\delta|} \geq \mathcal{O}(2^l \cdot \frac{|\mathcal{F}|}{|\mathcal{T}|})$. It should also be noted that $|\mathcal{T}|$ is already orders of magnitude smaller than $|\mathcal{F}|$ in most cases.

5.4 Error Bound of Support and Confidence of δ-TARs and Derived ARs

In this subsection, we analyze the error bound of the support and the confidence of a δ-TAR and of a derived AR. For clarity of presentation, up to now we have not distinguished between the exact and the estimated support and confidence of a δ-TAR/AR. To derive the error bound, we use *supp*(r) and *conf*(r) as the exact support and confidence of a δ-TAR/AR, r, and $\widetilde{supp}(r)$ and $\widetilde{conf}(r)$ as the estimated support and confidence of r. To avoid confusion, we may assume that the support and confidence of the δ-TARs/ARs discussed before this subsection are all estimated.

We first give the error bound of the estimated support and confidence of a δ-TAR.

Theorem 4 *Given $r = (X \Rightarrow (Y - X)) \in (\mathcal{R}_\delta - \mathcal{R}_\gamma)$. Let $\phi = (1 - \delta)^{|Y|-|X|}$. Then, the following expressions are true.*

$$(a) \qquad \frac{\widetilde{supp}(r) - supp(r)}{supp(r)} = 0\,,$$

$$(b) \qquad \begin{cases} \frac{\widetilde{conf}(r) - conf(r)}{conf(r)} = 0 & \text{if } X \in V_\mathcal{T}, \\ (\phi - 1) \leq \frac{\widetilde{conf}(r) - conf(r)}{conf(r)} \leq (\frac{1}{\phi} - 1) & \text{otherwise.} \end{cases}$$

Proof. Since $Y \in \mathcal{T}$, $\widetilde{supp}(Y) = supp(Y)$. Thus, $\widetilde{supp}(r) = supp(Y) = supp(r)$, from which follows the result of Part (a).

We now prove Part (b). If $X \in V_{\mathcal{T}}$, then $X \in \mathcal{T}$ and hence $\widetilde{supp}(X) = supp(X)$. Thus, $\widetilde{conf}(r) = \frac{supp(Y)}{\widetilde{supp}(X)} = \frac{supp(Y)}{supp(X)} = conf(r)$.

If $X \notin V_{\mathcal{T}}$, then $X \in V_b$. By Lemma 5 of (Cheng et al., 2006), $(\phi \cdot supp(X)) \leq \widetilde{supp}(X) \leq \frac{supp(X)}{\phi}$. Thus, $(\phi \cdot \frac{supp(Y)}{supp(X)}) \leq \frac{supp(Y)}{\widetilde{supp}(X)} \leq (\frac{1}{\phi} \cdot \frac{supp(Y)}{supp(X)})$. Since $conf(r) = \frac{supp(Y)}{supp(X)}$ and $\widetilde{conf}(r) = \frac{supp(Y)}{\widetilde{supp}(X)}$, we have $\phi \leq \frac{\widetilde{conf}(r)}{conf(r)} \leq \frac{1}{\phi}$. Hence, $(\phi - 1) \leq \frac{\widetilde{conf}(r) - conf(r)}{conf(r)} \leq (\frac{1}{\phi} - 1)$.

Next, we give the error bounds of the estimated support and confidence of a derived AR.

Theorem 5 *Given $r = (X \Rightarrow (Y - X)) \in \mathcal{R}$, where X, $Y \notin \mathcal{T}$. Let $\phi_s = (1-\delta)^i$ and $\phi_c = (1-\delta)^{i+j}$, where $i = (|Y'| - |Y|)$, $j = (|X'| - |X|)$, and X' and Y' are the closest δ-TCFI superset of X and Y. Then, the following expressions are true.*

$$(a) \qquad (\phi_s - 1) \leq \frac{\widetilde{supp}(r) - supp(r)}{supp(r)} \leq (\frac{1}{\phi_s} - 1) \ ,$$

$$(b) \qquad (\phi_c - 1) \leq \frac{\widetilde{conf}(r) - conf(r)}{conf(r)} \leq (\frac{1}{\phi_c} - 1) \ .$$

Proof. The error bound on the estimated support of r is the same as the error bound on the estimated frequency of Y defined in Lemma 5 of (Cheng et al., 2006). Thus, the result of Part (a) follows.

The estimated confidence of r is $\frac{\widetilde{supp}(Y)}{\widetilde{supp}(X)}$. Let $\phi = (1 - \delta)$. By Lemma 5 of (Cheng et al., 2006), we have $\frac{\phi^i \cdot supp(Y)}{supp(X)/\phi^j} \leq \frac{\widetilde{supp}(Y)}{\widetilde{supp}(X)} \leq \frac{supp(Y)/\phi^i}{\phi^j \cdot supp(X)}$. Thus, $(\phi^{i+j} \cdot \frac{supp(Y)}{supp(X)}) \leq \frac{\widetilde{supp}(Y)}{\widetilde{supp}(X)} \leq (\frac{1}{\phi^{i+j}} \cdot \frac{supp(Y)}{supp(X)})$, which is equivalent to $(\phi_c \cdot conf(r)) \leq \widetilde{conf}(r) \leq (\frac{1}{\phi_c} \cdot conf(r))$. The result of Part (b) thus follows.

Theorem 5 gives the worst-case bounds and does not consider the cases that $X \in \mathcal{T}$ and/or $Y \in \mathcal{T}$, for which the error bounds on the estimated support and confidence of r are much lower since $supp(X)$ and/or $supp(Y)$ are exact when $X \in \mathcal{T}$ and/or $Y \in \mathcal{T}$.

6 Constructing and Querying δ-TAR Tree

Given T_δ, it is straightforward to generate $\mathcal{R}_\delta$ as well as to derive $\mathcal{R}$. We can also recover $\mathcal{F}$ by traversing T_δ only once. In this section, we present an efficient algorithm to construct T_δ.

Algorithm 1 BuildTree

```
Input: The set of $\delta$-TCFIs $\mathcal{T}$.
Output: The $\delta$-TAR tree $T_\delta$.

1. Sort the $\delta$-TCFIs in $\mathcal{T}$ first in ascending order of their size,
     then in descending order of their support and finally in lexicographic order;
2. Create an empty Hashtable and an empty Inverted $\delta$-TCFI Index (ITI);
3. for each $X \in \mathcal{T}$ according to the sorted order do
4.       Create a new node $X$;
5.       Add $X$ to Hashtable and ITI;
6. Create the root, $\gamma$, of $T_\delta$;
7. for each $X \in \mathcal{T}$ do
8.       Find $X$'s parent, $Y$, and create the edge $(Y, X)$;
9.       $\mathcal{B} \leftarrow$ ComputeBorder($X$);
10.      for each $Z \in \mathcal{B}$ do
11.           Create a new node $Z$ and an edge $(X, Z)$;
12.           Add $Z$ to Hashtable;
13. return $T_\delta$;
```

The algorithm for constructing T_δ, *BuildTree*, is shown in Algorithm 1. Lines 3-4 create $V_\mathcal{T}$ and Line 6 creates the root γ. Then, Line 8 finds the parent of each vertex in $V_\mathcal{T}$ as defined in Definition 9 and creates $E_\mathcal{T}$ and E_γ. Line 9 computes the set of border vertices for each δ-TCFI and Line 11 creates V_b and E_b.

The algorithm is straightforward except Lines 8 and 9, which also determine the efficiency of the construction of T_δ. To efficiently process Lines 8 and 9, we use a hashtable, *Hashtable*, and an inverted index in the construction of T_δ. All vertices are hashed into Hashtable (Lines 5 and 12) based on the set of items of the FIs/δ-TCFIs, so that the vertices can be located in T_δ instantly. However, Lines 8 and 9 also involve searching the superset of a δ-TCFI and the closest δ-TCFI superset of an FI, which cannot be processed with a hashtable. For this purpose, we introduce an inverted index to facilitate the efficient processing of the search operation.

Definition 13 (INVERTED δ-TCFI INDEX) *The* Inverted δ-TCFI Index (*ITI*) *consists of the following components:*

– *An array, called the* Node-Link Array (NLA)*, stores the pointers to all δ-TCFIs in T_δ, except the size-1 δ-TCFIs (since they do not have non-empty subsets). The δ-TCFI whose pointer is stored in NLA[i] is assigned an* ID i.
– *An array, called the* Item Array (IA)*, stores the set of frequent items.*
– *Each item in the IA is associated with a list of arrays called* ID-arrays. *Each ID-array in the list stores a set of IDs, which are the IDs of those δ-TCFIs of the same size. Thus, an ID-array that stores the IDs of the size-n δ-TCFIs is called a* size-n ID-array.

The construction of the ITI is straightforward and efficient. Given $X \in \mathcal{T}$, where $X = x_1 \cdots x_n$, we add the pointer to the vertex X in T_δ to the end of the NLA. Then, we add X's ID to the end of the size-n ID-array of each item x_i, for $1 \le i \le n$. We can keep the IA in a hashtable so that we can access each x_i in the IA instantly.

The ID of X is simply assigned as the position of X's pointer stored in the NLA. Thus, the sorting of $\mathcal{T}$ performed at Line 1 of Algorithm 1 is to ensure an ordering on the IDs of the δ-TCFIs, so that the IDs in each ID-array are also automatically ordered. An example of an ITI is shown as follows.

Example 10. If $\delta = 0.03$, then $\mathcal{T} = \{$b, c, bd, cd, ac, bcd, acd, abcd$\}$ (sorted). Figure 4 shows the corresponding ITI. For example, the pointer of acd is stored at NLA[5]; thus, we have "5" in the size-3 ID-arrays of the items a, c and d, respectively. ■

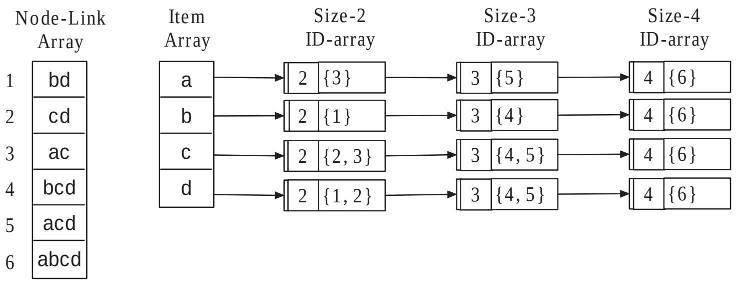


Fig. 4. Inverted δ-TCFI Index of Example 10

Given the ITI and $X \in \mathcal{F}$, where $X = x_1 \cdots x_n$, we can efficiently find X's superset in $\mathcal{T}$ that has the highest support among all X's supersets in $\mathcal{T}$. As shown in Procedure 1, we can find the superset of X by intersecting the size-j ID-arrays of each item x_i, where $j > n$. According to the way that $\mathcal{T}$ is sorted and that the ID of a δ-TCFI is added to the ID-arrays, the first ID obtained by the intersection of the size-j ID-arrays is X's size-j superset that has the greatest support and is lexicographically ordered before all other X's size-j supersets with the same support. X's closest δ-TCFI superset is the superset that gives X the greatest estimated support (Line 5 of Procedure 1).

Procedure 1 GetClosestSupset$(X = x_1 \cdots x_n)$

```
1. $Y \leftarrow \emptyset$;
2. for each $j = n+1, n+2, \ldots$ do
3.     Intersect the size-j ID-array of $x_i$,  for $1 \leq i \leq n$;
4.     $Y' \leftarrow NLA[ID]$, where $ID$ is the first ID obtained by the intersection (if any);
5.     if( $(supp(Y') \cdot ext(Y', j-n)) > (supp(Y) \cdot ext(Y, |Y|-n))$ )
6.         $Y \leftarrow Y'$;
7. return $Y$;
```

When X is a δ-TCFI, we can also find X's smallest superset that has the greatest support among all other X's smallest supersets (Line 8 of Algorithm 1) by simply returning the first δ-TCFI obtained by the intersection.

The following example further illustrates how Procedure 1 is processed.

Example 11. Referring to the ITI in Example 10, if we want to find the closest δ-TCFI superset of `ad`, we intersect the size-3 ID-arrays of `a` and `d` and obtain the ID "5". Thus, we access `acd` in T_δ via its pointer stored at Position 5 of the NLA. In the same way, we access `abcd` by intersecting the size-4 ID-arrays of `a` and `d`. Let N be the total number of transaction in the database we are referring to. Since $(supp(\texttt{acd}) \cdot ext(\texttt{acd}, 1)) = (104/N \times 1.03) = 107/N > (supp(\texttt{abcd}) \cdot ext(\texttt{abcd}, 2)) = (100/N \times 1.06) = 106/N$, `acd` is found to be the closest δ-TCFI superset of `ad`.

If we want to find the closest δ-TCFI superset of `ab`, we first intersect the size-3 ID-arrays of `a` and `b`. Since the intersection obtains no ID, we continue with the size-4 ID-arrays of `a` and `b` and obtain the ID "6". Thus, `abcd` is returned. ■

With Procedure 1, *ComputeBorder* (invoked at Line 9 of Algorithm 1), is presented in Procedure 2 as follows. For all $X \subset Y$ starting from $|X| = (|Y| - |EXT(Y)|)$, where $(|Y| - |EXT(Y)|)$ is the size of a smallest FI in $CLOS(Y)$, *GetClosestSupset* is invoked to determine whether Y is

Procedure 2 ComputeBorder(Y)

```
1. $\mathcal{B} \leftarrow \emptyset$;
2. for each $i = |EXT(Y)|, \ldots, 1$ do
3.     for each $X \subset Y$, where $|X| = (|Y| - i)$, do
4.         if($GetClosestSupset(X) = Y$ and $\nexists X' \in \mathcal{B}$ such that $X' \subset X$)
5.             $\mathcal{B} \leftarrow \mathcal{B} \cup \{X\}$;
6.     if($\nexists X \in \mathcal{B}$ such that $|X| = (|Y| - i)$)
7.         return $\mathcal{B}$;
```

the closest δ-TCFI superset of X. If Y is the closest δ-TCFI superset of X and X has no proper subset X' such that Y is also the closest δ-TCFI superset of X', then X is a border vertex and included in $\mathcal{B}$.

If no X, where $|X| = (|Y| - i)$, is a border vertex (Line 6), then X must have some proper subsets which are in $CLOS(Y)$. This also implies that X's supersets that are in $CLOS(Y)$ also have proper subsets which are in $CLOS(Y)$. Thus, no X's supersets can be border vertices and hence we can stop processing Y's subsets of size greater than $|X|$.

Example 12. Referring to the ITI in Example 10, we show how to compute the set of border vertices $\mathcal{B}$ of the δ-TCFI `abcd`. We start with the smallest subsets in $CLOS$(`abcd`), i.e., the size-2 subsets of `abcd` ($\delta = 0.03$). We find that `ac`, `bd` and `cd` are δ-TCFIs using Hashtable and that `ad` and `bc` are in $CLOS$(`acd`) and $CLOS$(`bcd`) using the ITI. Thus, only `ab` is included in $\mathcal{B}$. For the size-3 subsets of *abcd*, `acd` and `bcd` are found to be δ-TCFIs. The other two size-3 subsets of *abcd*, i.e., `abc` and `abd`, are also not included in $\mathcal{B}$, since they are supersets of `ab`, which is already in $\mathcal{B}$. Thus, only `ab` is returned as a border vertex of `abcd`. ■

6.1 Efficiency of ComputeBorder and GetClosestSupset

The bottom-up computation for the border vertices in ComputeBorder terminates in two or three levels in most cases. Since the number of border vertices is small, the cost of the subset testing in Line 4 of Procedure 2 is also small, while we keep the itemsets in $\mathcal{B}$ using a bit array to allow efficient subset testing. Thus, the most costly operation is still GetClosestSupset.

The efficiency of GetClosestSupset depends on the size of the ID-arrays to be intersected. The size of an ID-array is small in most cases, because the number of δ-TCFIs is small and each ID-array consists of only a local set of δ-TCFI supersets of an item of a specific size. Since the ID-arrays are ordered, we can employ binary search, instead of linear scan, for matching the IDs during the intersection. In our implementation, binary search is used only when the size of an ID-array is larger than a preset threshold. Moreover, since each intersection only needs to return the first ID, the process terminates early in most cases. The intersection also terminates when the end of any ID-array is reached. Thus, we can speed up the process by first sorting the ID-arrays to be intersected in ascending order of their size.

6.2 Querying δ-TAR Tree

In practice, $\mathcal{F}$ and $\mathcal{R}$ are too large to be generated, to be kept in disk, and to be analyzed. Given $\mathcal{T}$ and $\mathcal{R}_\delta$, applications may not actually demand to recover the whole sets of $\mathcal{F}$ and $\mathcal{R}$. Instead, only a small portion of relevant FIs/ARs is required to be recovered each time. The δ-TAR tree T_δ offers a practical way of generating and analyzing the FIs/ARs on demand: given some δ-TCFIs/δ-TARs of interest, we can efficiently retrieve/derive relevant patterns and rules from T_δ for further analysis. In cases that an application demands the whole sets of ARs/FIs, we can also generate them from T_δ. The δ-TAR tree, together with its two supporting structures, Hashtable and the ITI, provides very efficient and effective retrieval of relevant patterns and rules and allows the user to analyze the mining results in a more interactive way.

We note that the ITI can also be used to obtain all the subsets and supersets of an FI. Efficient processing of this operation is important for retrieving useful information such as, given $X \in \mathcal{F}$, find the set of ARs in which X is the antecedent or the consequent. We can process this by modifying slightly Procedure 1. To obtain all δ-TCFI supersets of X, we do not terminate after we obtain the first ID but continue until the end of an ID-array is reached. To obtain all δ-TCFI subsets of X, for each $j < |X|$, the intersection returns the IDs that appear in the size-j ID-arrays of j items in X. Then, we can access the δ-TCFIs to derive other supersets and subsets of X that are not δ-TCFIs.

7 Experimental Evaluation

In this section, we verify the effectiveness of δ-TARs as a concise representation of ARs, as well as the efficiency of our algorithm for mining δ-TARs. We run all experiments on an AMD Opteron 248 with 8GB RAM, running Linux 64-bit.

Datasets. We conduct our experiments on the popularly used real datasets from (FIMI Dataset Repository, 2003). We choose three datasets with the following representative characteristics, which are true for a wide range of values of σ.

– `pumsb*`: the number of CFIs is orders of magnitude smaller than that of FIs, but is orders of magnitude larger than that of MFIs.
– `accidents` (Geurts et al., 2003): the number of CFIs is almost the same as that of FIs, and is orders of magnitude larger than that of MFIs.
– `mushroom`: the number of CFIs is orders of magnitude smaller than that of FIs, but is only a few times larger than that of MFIs.

Among the real datasets, `pumsb*` and `mushroom` are dense datasets while `accidents` is a sparser dataset. Other information of the datasets is shown in Table 1.

Dataset Name	Number of Unique Items	Number of Transactions	Max. Size of A Transaction	Avg. Size of A Transaction
`pumsb*`	7,117	49,046	63	50
`accidents`	469	340,183	51	34
`mushroom`	120	8,124	23	23

Table 1. Real and Synthetic Datasets

Experimental Settings. For all the experiments, we use our algorithm BuildTree to first construct the δ-TAR tree T_δ and then generate the set of δ-TARs from T_δ. The input to BuildTree, i.e., the set of δ-TCFIs, is computed by the algorithm MineTCFI proposed in (Cheng et al., 2006). We set δ to be 0.05, 0.05 and 0.2 for `pumsb*`, `accidents` and `mushroom`, respectively, as recommended by (Cheng et al., 2006).

We compare our δ-TARs with the following state-of-the-art concise representations of ARs: *Closed ARs* (Zaki, 2004), *Basic ARs* (Li and Hamilton, 2004), and *Non-Derivable ARs* (*NDARs*) (Goethals et al., 2005).

7.1 Performance at Different Minimum Confidence Thresholds

We first report the results for the different values of ς, from 0 to 1.

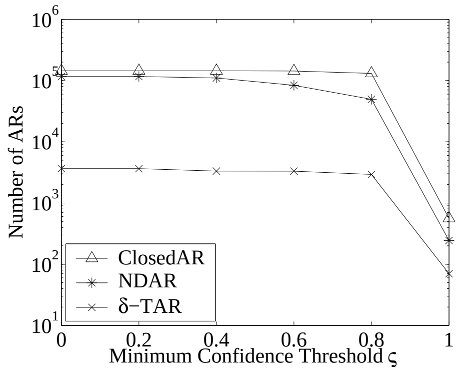


Fig. 5. Number of Rules at Varying ς (`pumsb*` at $\sigma = 0.3$)

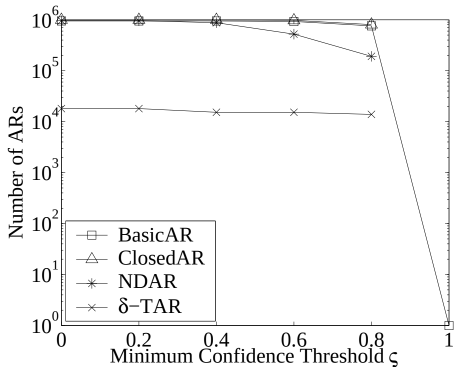


Fig. 6. Number of Rules at Varying ς (`accidents` at $\sigma = 0.3$)

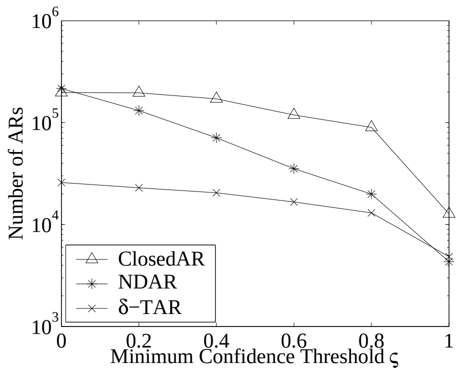


Fig. 7. Number of Rules at Varying ς (`mushroom` at $\sigma = 0.03$)

Number of Rules. Figures 5 to 7 report the number of rules returned by each of the approaches. The results show that the number of δ-TARs is almost two orders of magnitude smaller than the number of Closed

ARs, Basic ARs and NDARs for both `pumsb*` and `accidents`, and is up to an order of magnitude smaller for `mushroom`. For `accidents` at $\varsigma = 1$ as shown in Figure 6, we do not record any δ-TAR, Closed AR, or NDAR, although there are 16 ARs of confidence 1, which can be derived from other ARs. For reference, we remark that the number of δ-TARs is approximately six, three, and six orders of magnitude smaller than the number of all ARs, for the three datasets `pumsb*`, `accidents`, and `mushroom`, respectively.

We are not able to obtain the Basic ARs for both `pumsb*` and `mushroom`. However, as reported in (Goethals et al., 2005), the number of Basic ARs is larger than that of NDARs for the same set of datasets used (but at much larger σ). Thus, we believe the number of Basic ARs is also orders of magnitude larger than that of δ-TARs. We report the number of Basic ARs for `accidents` in Figure 6 as a reference.

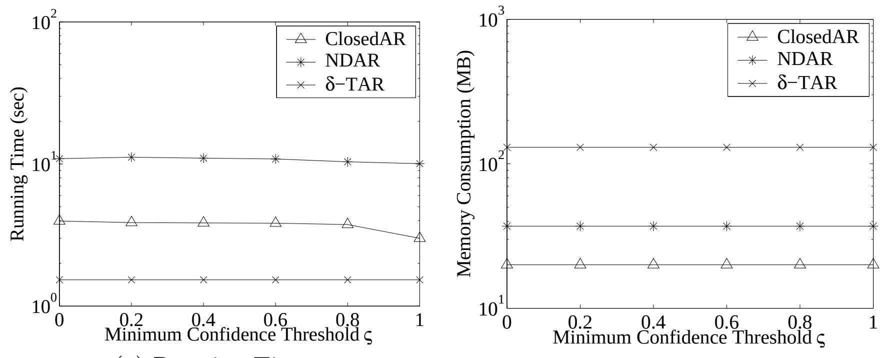
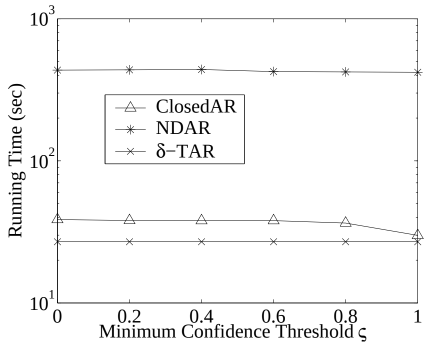


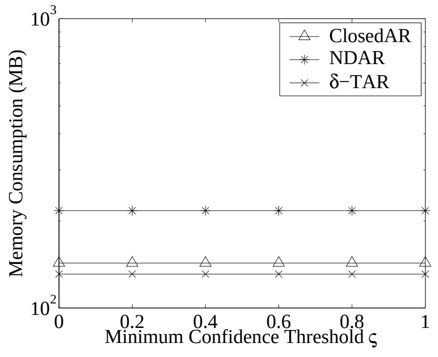
Fig. 8. Running Time and Memory Consumption at Varying ς (`pumsb*` at $\sigma = 0.3$)

Running Time and Memory Consumption. Figures 8 to 10 report the time and memory consumption for generating the respective rules. Since Closed ARs and NDARs are generated from the raw datasets directly instead of from the CFIs and NDIs, we also include the time and memory used to mine the δ-TCFIs.

As shown in the figures, there is no clear winner for both running time and memory usage. Mining δ-TARs is the fastest for `pumsb*` and `accidents` but is the slowest for `mushroom`, while the memory consumption of mining δ-TARs is the least for `accidents` but is the greatest for `pumsb*` and `mushroom`. However, the difference in the running time and the memory consumption is small and all the three algorithms are effi-

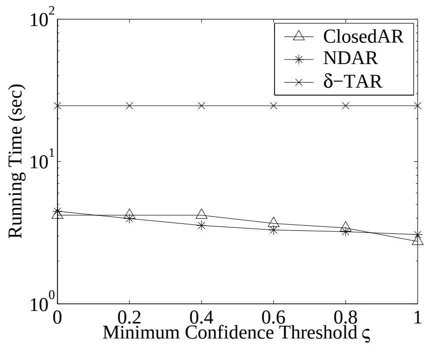


(a) Running Time

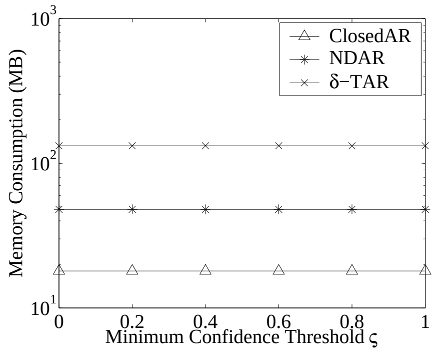


(b) Memory Consumption

Fig. 9. Running Time and Memory Consumption at Varying ς (`accidents` at $\sigma = 0.3$)



(a) Running Time



(b) Memory Consumption

Fig. 10. Running Time and Memory Consumption at Varying ς (`mushroom` at $\sigma = 0.03$)

cient in general. We also find that the time taken to generate the δ-TARs is roughly proportional to the number of δ-TCFIs, which is the input to build T_δ. Since we first build T_δ and then generate the δ-TARs, the time and the memory used for the different settings of ς is basically the same.

Error Rate. We also use T_δ to derive all ARs of confidence greater than 0.6 for each of the three datasets. We then compute the error rate of the confidence and support of the derived ARs as (|*true conf* (or *supp*) − *estimated conf* (or *supp*)|)/(*true conf* (or *supp*)). The error rates for the individual ARs are then averaged. The average error rates of the estimated confidence (support) of the derived ARs are computed to be 0.09 (0.02), 0.07 (0.03) and 0.07 (0.02) for `pumsb*`, `accidents` and `mushroom`, respectively. The error rate of the estimated confidence is greater than that of the estimated support because the computation of confidence is based on the support of the antecedent and the consequent of an AR, both of which may be estimated.

7.2 AR Derivation and Querying δ-TAR Tree

We compute the average time (i.e., total time divided by the number of ARs derived) taken to derive an AR, which is $5 \times 10^{-12} sec$, $56 \times 10^{-12} sec$ and $8 \times 10^{-12} sec$ for `pumsb*`, `accidents` and `mushroom`, respectively. The time is longer for `accidents` because `accidents` is sparser and hence more traversals on T_δ are needed to derive the ARs.

To show the efficiency of querying T_δ, we also compute the time for finding all ARs in which X is the antecedent or the consequent, for each $X \in \mathcal{F}$. The average time taken to process each FI is recorded as $0 sec$ for all the three datasets. The result is surprising since a set of ARs is actually returned for each FI, but the processing time is smaller than that of deriving a single AR tested earlier. The main reason for the shorter running time is because we do not need to traverse T_δ to find the subsets and supersets of X to determine whether X is the antecedent or the consequent; instead, we can very efficiently obtain the subsets and supersets of X using the ITI and then access the corresponding nodes in T_δ directly. The result of this experiment thus reveals the efficiency of using the ITI and querying T_δ.

7.3 Performance at Different Minimum Support Thresholds

This experiment studies the effect of different values of σ on the performance of mining δ-TARs. We fix $\varsigma = 0$ and vary σ for each dataset as shown in Figures 11 to 13, which report the number of the three types of rules. The figures clearly show that for all settings of σ and all datasets, the number of δ-TARs is consistently one or nearly two orders of magnitude smaller than that of Closed ARs and NDARs.

The running time and memory consumption recorded for this experiment is consistent with the trend observed in the experiment in Section 7.1 and in overall, all the three approaches are efficient and there is no clear winners. Same as the previous experiment, the running time and memory consumption of mining δ-TARs is roughly proportional to the number of δ-TCFIs. Thus, we omit the detailed figures.

8 Conclusions

In this paper, we study an effective technique for eliminating the redundancy in the set of ARs. We propose a concise set of ARs, called the

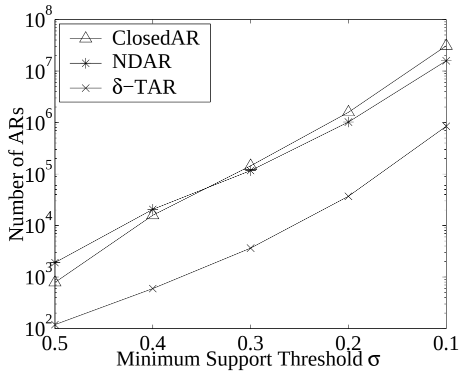


Fig. 11. Number of Rules at Varying σ (`pumsb*`)

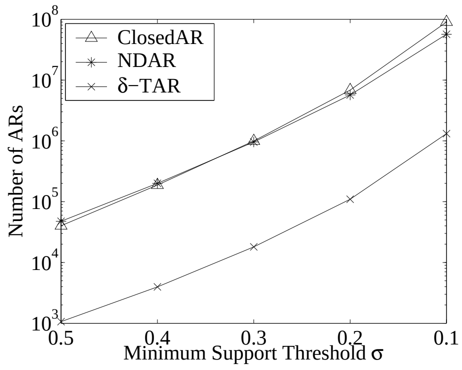


Fig. 12. Number of Rules at Varying σ (`accidents`)

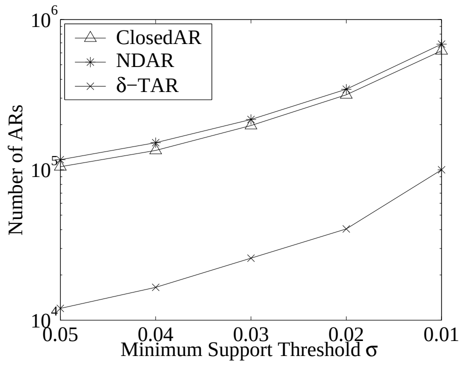


Fig. 13. Number of Rules at Varying σ (`mushroom`)

δ-TARs, which we prove to be non-redundant and able to derive all ARs by three inference rules. In addition, we develop the δ-TAR tree that provides a practical way of allowing users to query the mining results and derive relevant knowledge on demand. Experiments verify that the set of δ-TARs is up to orders of magnitude smaller than the state-of-the-art approaches (Zaki, 2004; Li and Hamilton, 2004; Goethals et al., 2005). We also show that the δ-TAR tree is a very efficient supporting structure for generating the set of δ-TARs, deriving the ARs and querying the δ-TCFIs and FIs.

For future work, we seek to optimize the performance of AR querying by considering a caching system (Jeudy and Boulicaut, 2002), as well as to further reduce the number of rules by pruning the non-actionable ARs (Liu et al., 2001).

Acknowledgements. The authors thank Dr. Bart Goethals and Mr. Juho Muhonen for providing us the algorithm for mining NDARs, Prof. Howard Hamilton and Mr. Guichong Li for providing us the algorithm for mining Basic ARs, and Prof. Mohammed Javeed Zaki for providing us the algorithm for mining Closed ARs.

Bibliography

Charu C. Aggarwal and Philip S. Yu. A new approach to online generation of association rules. *IEEE Transactions on Knowledge and Data Engineering*, 13(4):527–540, 2001.

Rakesh Agrawal, Tomasz Imielinski, and Arun N. Swami. Mining association rules between sets of items in large databases. In *Proc. of the ACM Conference on the Management of Data (SIGMOD)*, 1993.

Yves Bastide, Nicolas Pasquier, Rafik Taouil, Gerd Stumme, and Lotfi Lakhal. Mining minimal non-redundant association rules using frequent closed itemsets. In *Computational Logic*, pages 972–986, 2000.

Roberto Bayardo. Efficiently mining long patterns from databases. In *Proc. of the ACM Conference on the Management of Data (SIGMOD)*, pages 85–93, 1998.

Jean-François Boulicaut, Artur Bykowski, and Christophe Rigotti. Free-sets: A condensed representation of boolean data for the approximation of frequency queries. *Data Mining and Knowledge Discovery (DMKD)*, 7(1):5–22, 2003.

Toon Calders and Bart Goethals. Mining all non-derivable frequent itemsets. In *Proc. of the European Conference on Principles of Data Mining and Knowledge Discovery (PKDD)*, pages 74–85, 2002.

Aaron Ceglar and John F. Roddick. Association mining. *ACM Computing Surveys (CSUR)*, 38(2):5, 2006.

James Cheng, Yiping Ke, and Wilfred Ng. δ-Tolerance Closed Frequent Itemsets. *In Proc. of the 6th IEEE International Conference on Data Mining (ICDM)*, 2006.

James Cheng, Yiping Ke, and Wilfred Ng. FG-Index: Towards verification-free query processing on graph databases. *To appear in Proc. of the 26th ACM Conference on the Management of Data (SIGMOD)*, 2007a.

James Cheng, Yiping Ke, and Wilfred Ng. Maintaining frequent closed itemsets over a sliding window. *To appear in Journal of Intelligent Information Systems (JIIS)*, 2007b.

FIMI Dataset Repository. The FIMI frequent itemset mining dataset repository. *http://fimi.cs.helsinki.fi/data/*, 2003.

Bruno M. Fonseca, Paulo Braz Golgher, Bruno Pôssas, Berthier A. Ribeiro-Neto, and Nivio Ziviani. Concept-based interactive query expansion. In *Proc. of the ACM CIKM International Conference on Information and Knowledge Management*, pages 696–703, 2005.

Karolien Geurts, Geert Wets, Tom Brijs, and Koen Vanhoof. Profiling high frequency accident locations using association rules. In *Proc. of the 82nd Annual Transportation Research Board*, page 18, 2003.

Bart Goethals, Juho Muhonen, and Hannu Toivonen. Mining non-derivable association rules. In *Proc. of the SIAM International Conference on Data Mining (SDM)*, 2005.

Baptiste Jeudy and Jean-François Boulicaut. Using condensed representations for interactive association rule mining. In *Proc. of the 6th European conferences on Principles and practice of Knowledge Discovery in Databases (PKDD)*, pages 225–236, 2002.

Marzena Kryszkiewicz. Representative association rules and minimum condition maximum consequence association rules. In *Proc. of the European Conference on Principles of Data Mining and Knowledge Discovery (PKDD)*, pages 361–369, 1998.

Navin Kumar, Aryya Gangopadhyay, and George Karabatis. Supporting mobile decision making with association rules and multi-layered caching. *Decision Support Systems*, 43(1):16–30, 2007.

Guichong Li and Howard J. Hamilton. Basic association rules. In *Proc. of the SIAM International Conference on Data Mining (SDM)*, 2004.

Bing Liu, Wynne Hsu, and Yiming Ma. Identifying non-actionable association rules. In *Proc. of the 7th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2001.

Fabian Mörchen and Alfred Ultsch. Efficient mining of understandable patterns from multivariate interval time series. *To appear in Data Mining and Knowledge Discovery (DMKD)*, 2007.

Girish K. Palshikar, Mandar S. Kale, and Manoj M. Apte. Association rules mining using heavy itemsets. *Data and Knowledge Engineering*, 61(1):93–113, 2007.

Nicolas Pasquier, Yves Bastide, Rafik Taouil, and Lotfi Lakhal. Discovering frequent closed itemsets for association rules. In *Proc. of the International Conference on Database Theory (ICDT)*, 1999.

Nicolas Pasquier, Rafik Taouil, Yves Bastide, Gerd Stumme, and Lotfi Lakhal. Generating a condensed representation for association rules. *Journal of Intelligent Information Systems (JIIS)*, 24(1):29–60, 2005.

F. A. Thabtah and P. I. Cowling. A greedy classification algorithm based on association rule. *Applied Soft Computing*, 7(3):1102–1111, 2007.

Qiang Yang, Tianyi Li, and Ke Wang. Building association-rule based sequential classifiers for web-document prediction. *Data Mining and Knowledge Discovery (DMKD)*, 8(3):253–273, 2004.

Mohammed Javeed Zaki. Mining non-redundant association rules. *Data Mining and Knowledge Discovery*, 9(3):223–248, 2004.