Online Association Rule Mining *

Christian Hidber
International Computer Science Institute, Berkeley
hidber@icsi.berkeley.edu

May 20, 1998

Abstract

We present a novel algorithm to compute large itemsets online. It needs at most two scans of the transaction sequence. Any time during the first scan, the user is free to change the support threshold. The algorithm maintains a superset of all large itemsets and a deterministic lower and upper bound on the support of each itemset. We continously display the resulting association rules along with an interval on the rule's support and confidence. The algorithm can compute association rules for a transaction sequence which is read from a network and is too large to be stored locally for a rescan. During the second scan we determine the precise support for each large itemset and prune all small itemsets using a new forward-pruning technique.

1 Introduction

Mining for association rules is a form of data mining introduced in [AIS93]. The prototypical example is based on a list of purchases in a store. An association rule for this list is a rule such as "85% of all customers who buy product A and B also buy product C and D". Discovering such customer buying patterns is useful for customer segmentation, cross-marketing, catalog design and product placement.

*Technical Report, UCB//CSD-98-1004, Department of Electrical Engineering and Computer Science, University of California at Berkeley

We give a problem description which follows [BMUT97]. The *support* of an itemset (set of items) in a transaction sequence is the fraction of all transactions containing the itemset. An itemset is called *large* if its support is greater or equal to a user-specified *support threshold*, otherwise it is called *small*. An *association rule* is an expression $X \Rightarrow Y$ where X and Y are disjoint itemsets. The *support* of this rule is the support of $X \cup Y$. The *confidence* of this rule is the fraction of all transactions containing X that also contain Y, i.e. the support of $X \cup Y$ divided by the support of X. In the example above, the "85%" is the confidence of the rule $\{A, B\} \Rightarrow \{C, D\}$. For an association rule to hold, it must have a support $\geq$ a user-specified support threshold and a confidence $\geq$ a user-specified confidence threshold. Existing algorithms proceed in 2 steps to compute association rules:

1. Find all large itemsets.

2. For each large itemset Z, find all subsets X, such that the confidence of $X \Rightarrow Z \backslash X$ is greater or equal to the confidence threshold.

We address the first step, since the second step can already be computed online, c.f. [AY97]. Existing large itemset computation algorithms have an offline or batch behaviour: given the user-specified support threshold, the transaction sequence is scanned and rescanned, often several times, and eventually all large itemsets are produced. However, the user does not know, in general, an appropriate support threshold in advance. An inappropriate choice yields, after a long wait, either too many or too few large itemsets, which often results in useless or misleading association rules.

Inspired by online aggregation, c.f. [Hel96, HHW97], our goal is to overcome these difficulties by bringing large itemset computation online. We consider an algorithm to be online if: *1)* it gives continuous feedback, *2)* it is user controllable during processing and *3)* it yields a deterministic and accurate result. Random sampling algorithms produce results which hold with some probability < 1. Thus we do not view them as being online.

In order to bring large itemset computation online, we introduce a novel algorithm called CARMA (Continuous Association Rule Mining Algorithm). The algorithm needs, at most, two scans of the transaction sequence to produce all large itemsets.

During the first scan, the algorithm continuously constructs a lattice of all potentially large itemsets (large with respect to the scanned part of the transaction sequence). For each set in the lattice, CARMA provides a deterministic lower and upper bound for its support. We continuously display, e.g. after each transaction processed, the resulting association rules to the user along with bounds on each rule's support and confidence. The user is free to adjust the support and confidence thresholds at any time. Adjusting the support threshold may result in an increased threshold for which the algorithm guarantees to include all large itemsets in the lattice. If satisfied with the rules and bounds produced so far, the user can stop the rule mining early.

During the second scan, the algorithm determines the precise support of each set in the lattice and continuously removes all small itemsets. While requiring, at most, two scans of the transaction sequence, CARMA typically needs less than two scans using a novel "forward pruning" technique.

Existing algorithms need to rescan the transaction sequence. Thus, they can not be used on a sequence which is read from a network and cannot feasably be stored locally, e.g. real-time sequences read from the internet. In contrast, using CARMA's first-scan algorithm, we can read such a transaction sequence from a network and continuously generate the resulting association rules online, not requiring a rescan.

2 Overview

The paper is structured as follows: In Section 3, we put our algorithm in the context of related work. In Section 4, we give a sketch of CARMA. It uses two distinct algorithms for the first and second scan, called *PhaseI* and *PhaseII* respectively. In Section 5, we describe PhaseI in detail, illustrating the algorithm in an example in Subsection 5.3 and discuss changing support thresholds in Subsection 5.4. We conclude the section with remarks on pruning strategies in Subsection 5.5. In Section 6, we introduce the PhaseII algorithm for the second scan. After a detailed description of our forward pruning technique in Subsection 6.2, we give the formal definition of PhaseII in Subsection 6.3. In Section 7, we put the PhaseI and PhaseII algorithms together, yielding CARMA. In Appenix A and Appenix B we give the formal proofs of correctness for PhaseI and PhaseII.

3 Related Work

Most large itemset computation algorithms are related to the *Apriori* algorithm due to Agrawal & Srikant, c.f. [AS94]. See [AY98] for a survey of large itemset computation algorithms. Apriori exploits the observation that all subsets of a large itemset are large themselves. It is a multi-pass algorithm, where in the k-th pass all large itemsets of cardinality k are computed. Hence Apriori needs up to $c+1$ scans of the database where c is the maximal cardinality of a large itemset.

In [SON95] a 2-pass algorithm called *Partition* is introduced. The general idea is to partition the database into blocks such that each block fits into main-memory. In the first pass, each block is loaded into memory and all large itemsets, with respect to that block, are computed using Apriori. Merging all resulting sets of large itemsets then yields a superset of all large itemsets. In the second pass, the actual support of each set in the superset is computed. After removing all small itemsets, Partition produces the set of all large itemsets.

In contrast to Apriori, the DIC (Dynamic Itemset Counting) algorithm counts itemsets of different cardinality simultaneously, c.f. [BMUT97]. The transaction sequence is partioned into blocks. The itemsets are stored in a lattice which is initialized by all singleton sets. While a block is scanned, the count (number of occurences) of each itemset in the lattice is adjusted. After a block is processed, an itemset is added to the lattice if and only if all its subsets are potentially large, i.e. large with respect to the part of the transaction sequence for which its count was maintained. At the end of the sequence, the algorithm rewinds to the beginning. It terminates when the count of each itemset in the lattice is determined. Thus after a finite number of scans, the lattice contains a superset of all large itemsets and their counts. For suitable block sizes, DIC requires fewer scans than Apriori.

We note that all of the above algorithms: *1)* require that the user specifies a fixed support threshold in advance, *2)* do not give any feedback to the user while they are running and *3)* may need more than two scans (except Partition). CARMA, in contrast: *1)* allows the user to change the support threshold at any time, *2)* gives continuous feedback and *3)* requires at most two scans of the transaction sequence.

Random sampling algorithms have been suggested as well, c.f. [Toi96, ZPLO96]. The general idea is to take a random sample of suitable size from the transaction sequence and compute the large itemsets using Apriori or Partition with respect to that sample. For each itemset, an interval is computed such that the support lies within the interval with probability $\geq$ some threshold. CARMA, in contrast, deterministically computes all large itemsets along with the precise support for each itemset.

Several algorithms based on Apriori were proposed to update a previously computed set of large itemsets due to insertion or deletion of transactions, c.f. [CHNW96, CLK97, TBAR97]. These algorithms require a rescan of the full transaction sequence whenever an itemset becomes large due to an insertion. CARMA, in contrast, requires a rescan only if the user needs, instead of the supplied deterministic intervals, the precise support of the additional large itemsets.

In [AY97] an Online Analytical Processing (OLAP)-style algorithm is proposed to compute association rules. The general idea is to precompute all large itemsets relative to some support threshold s using a traditional algorithm. The association rules are then generated online relative to an interactively specified confidence threshold and support threshold $\geq s$. We note that: *1)* the support threshold s must be specified before the precomputation of the large itemsets, *2)* the large itemset computation remains offline and *3)* only rules with support $\geq s$ can be generated. CARMA overcomes these difficulties by bringing the large itemset computation itself online. Thus, combining CARMA's large itemset computation with the online rule generation suggested in [AY97] brings both steps online, not requiring any precomputation.

4 Sketch of the Algorithm

CARMA uses distinct algorithms, called PhaseI and PhaseII, for the first and second scan of the transaction sequence. In this section, we give a sketch of these algorithms. For a detailed description and formal definition see Section 5, Section 6 and Section 7.

During the first scan PhaseI continuously constructs a lattice of all potentially large itemsets. After each transaction, it inserts and/or removes some itemsets from the lattice. For each itemset v, PhaseI stores the following three integers (see Figure 1 below, the itemset $\{a, b\}$ was inserted in the lattice while reading the j-th transaction, the current transaction index is i):

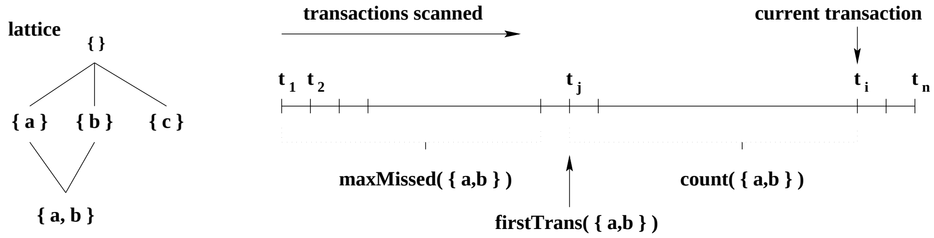

Figure 1

count(v)	the number of occurences of v *since* v was inserted in the lattice.
firstTrans(v)	the index of the transaction *at* which v was inserted in the lattice.
maxMissed(v)	upper bound on the number of occurences of v *before* v was inserted in the lattice.

For each transaction read, we increment the count of all itemsets contained in the current transaction. Suppose we are reading transaction i. For any itemset v in the lattice, we get a deterministic lower bound $count(v)/i$ and upper bound $(maxMissed(v) + count(v))/i$ on its support in the first i transactions. We denote these bounds by $minSupport(v)$ and $maxSupport(v)$ respectively. The computation of $maxMissed(v)$ during the insertion of v in the lattice is a central part of the algorithm. It not only depends on v and the current transaction index but also on the previous values of the support threshold, since the user may change the support threshold at any time. At the end of the transaction sequence, the lattice contains a superset of all large itemsets. We then rewind to the beginning and start PhaseII.

PhaseII initially removes all itemsets which are trivially small, i.e. itemsets with $maxSupport$ below the current user-specified support threshold. By rescanning the transaction sequence, PhaseII determines the precise number of occurences of each remaining itemset and continuously removes all small itemsets using our forward-pruning technique. Eventually, we end up with the set of all large itemsets along with their support.

5 PhaseI Algorithm

In this section, we fully describe the PhaseI algorithm, which constructs a superset of all large itemsets. In Subsection 5.1, we introduce the concept of a *support lattice* and of a *support sequence*, which we need to define PhaseI in Subsection 5.2. In Subsection 5.3 we illustrate the algorithm in a simple example. In Subsection 5.4, we discuss changing support thresholds and in Subsection 5.5, admissible pruning strategies.

5.1 Support Lattice & Support Sequence

For a given transaction sequence and an itemset v, we denote by $support_i(v)$ the support of v in the first i transactions. Let V be a lattice of itemsets such that along with each itemset $v \in V$ we have the three associated integers $count(v)$, $firstTrans(v)$ and $maxMissed(v)$ as defined in Section 4. We call V a *support lattice* (up to i and relative to the support threshold s) if and only if V contains all itemsets v with $support_i(v) \geq s$. Hence, a support lattice is a superset of all large itemsets in the first i transactions with respect to the support threshold s.

For each transaction processed, the user can specify an arbitrary support threshold. We get a sequence of support thresholds $\sigma = (\sigma_1, \sigma_2, \ldots)$ where σ_i denotes the support threshold for the i-th transaction. We call such a sequence a *support sequence*. For a support sequence σ and an integer i, we denote by $\lceil \sigma \rceil_i$ the least monotone decreasing sequence which is up to i pointwise greater or equal to σ and 0 otherwise (see Figure 2 below).

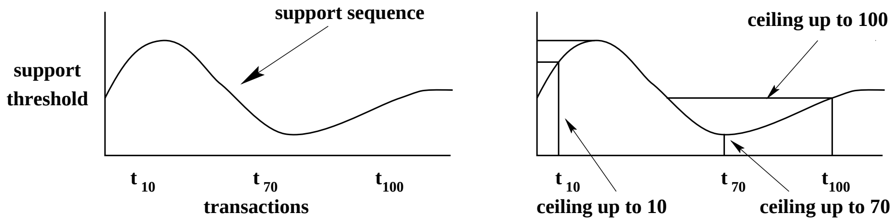

Figure 2

We call $\lceil \sigma \rceil_i$ the *ceiling of* σ *up to* i. By $avg_i(\sigma)$ we denote the running average of σ up to i, i.e. $avg_i(\sigma) = \frac{1}{i} \sum_{j=1}^{i} \sigma_j$.

5.2 PhaseI Algorithm

In this subsection, we give a full description and formal definition of the PhaseI algorithm, which constructs a superset of all large itemsets. Since a support lattice is a superset of all large itemsets, it suffices for PhaseI to maintain a support lattice V while scanning the transaction sequence:

We initialize V to $\{\emptyset\}$ and set

$$count(\emptyset) = 0,\ firstTrans(\emptyset) = 0 \text{ and } maxMissed(\emptyset) = 0.$$

Thus V is a support lattice for an empty transaction sequence. Suppose V is a support lattice up to transaction $i-1$, we are reading the i-th transaction t_i and we want to transform V into a support lattice up to i. Let σ_i be the current user-specified support threshold. To maintain the lattice we proceed in three steps: *1) increment* the count of all itemsets occuring in the current transaction, *2) insert* some itemsets in the lattice and *3) prune* some itemsets from the lattice.

1) Increment: We increment $count(v)$ for all itemsets $v \in V$ that are contained in t_i, maintaining the correctness of all integers stored in V.

2) Insert: We insert a subset v of t_i in V if and only if all subsets w of v are already contained in V and are potentially large, i.e. $maxSupport(w) \geq \sigma_i$. This corresponds to the observation that the set of all large itemsets is closed under subsets. This condition also limits the growth of V, since only minimal supersets of sets in V are added. Thus, the maximal cardinality of all sets in V increases at most by 1 per transaction processed. Inserting v in V, we set $firstTrans(v) = i$ and $count(v) = 1$, since v is contained in the current transaction t_i. Since $support_i(w) \geq support_i(v)$ for all subsets w of v and $w \subset t_i$ we get

$$maxMissed(v) \leq maxMissed(w) + count(w) - 1.$$

In the following Theorem 1 we prove that

$$support_{i-1}(v) > \max\{\, avg_{i-1}(\lceil\sigma\rceil_{i-1}),\ \tfrac{|v|-1}{i-1}\,\} \quad \text{implies} \quad v \in V.$$

Since we are inserting v in V, the set v is not contained in V yet. Hence, we get from Theorem 1

$$support_{i-1}(v) \leq \max\{\, avg_{i-1}(\lceil\sigma\rceil_{i-1}),\ \tfrac{|v|-1}{i-1}\,\}.$$

For a real number x we denote by $\lfloor x \rfloor$ the largest integer less or equal to x, i.e. $\lfloor x \rfloor = \max\{i \in \mathbb{Z} \mid x \geq i\}$. Since $maxMissed(v)$ is an integer we get

$$maxMissed(v) \leq \max\{\, \lfloor (i-1)avg_{i-1}(\lceil\sigma\rceil_{i-1}) \rfloor,\ |v| - 1\,\}.$$

Thus we define $maxMissed(v)$ as

$$\begin{aligned} \min \quad \{ \quad & \max\{ \lfloor (i-1) avg_{i-1}(\lceil \sigma \rceil_{i-1}) \rfloor, |v|-1 \}, \\ & maxMissed(w) + count(w) - 1 \mid w \subset v \quad \}. \end{aligned} \tag{1}$$

In particular we get $maxMissed(v) \leq i-1$, since the emptyset is a subset of v, $\emptyset$ is an element of V and the count of $\emptyset$ equals i, the current transaction index.

3) Prune: We do not prescribe a particular pruning strategy. However we require that only small itemsets, i.e. $maxSupport(v) < \sigma_i$, are removed from V and if a set is removed then all its supersets are removed as well. See Subsection 5.5 for further remarks on pruning strategies.

We now have the following algorithm:

Function PhaseI(transaction sequence $(t_1, \ldots, t_n)$,
  support sequence $\sigma = (\sigma_1, \ldots, \sigma_n)$) : support lattice;
support lattice V;
begin
  $V := \{\emptyset\}$, $maxMissed(v) := 0$, $firstTrans(v) := 0$ $count(v) := 0$.
  for i from 1 to n do
    Increment: for all $v \in V$ with $v \subseteq t_i$ do $count(v)++$;
    Insert: for all $v \subseteq t_i$ with $v \notin V$ do
      if $\forall w \subset v : w \in V$ and $maxSupport(w) \geq \sigma_i$ then
        $V := V \cup \{v\}$;
        $maxMissed(v) :=$
          $\min\{ \max\{ \lfloor (i-1) avg_{i-1}(\lceil \sigma \rceil_{i-1}) \rfloor, |v|-1 \},$
            $maxMissed(w) + count(w) \mid w \subset v \}$;
        $firstTrans(v) := i$;
        $count(v) := 1$;
      fi; od;
    Prune: remove $v \in V$ from V only if $maxSupport(v) < \sigma_i$.
      if $v \in V$ is removed, remove all supersets as well.
  od;
  return V;
end;

Figure 3

We omitted any optimization in the above definition of PhaseI. For example, the incrementation and insertion step can be accomplished by traversing the support lattice once. As another example, it suffices for the *if* clause as well as for $maxMissed$ computation to check subsets of v of cardinality $|v|-1$ instead of all subsets.

The following theorem asserts the correctness of the algorithm:

Theorem 1 *Let V be the lattice returned by $PhaseI(T, \sigma)$ for a transaction sequence T of length n and support sequence σ.*

Then V is a support lattice relative to the suport threshold

$$\max\{\, avg_n(\lceil\sigma\rceil_n),\ \frac{c+1}{n}\,\}$$

with c the maximal cardinality of a large itemset in T. For any itemset v

$$support_n(v) \geq \max\{\, avg_n(\lceil\sigma\rceil_n),\ \frac{|v|-1}{n}\,\} \quad implies \quad v \in V.$$

Proof: see Theorem 4 in Appenix A. □

For a disccussion of the involved support thresholds see Subsection 5.4.

5.3 Example

We illustrate in this subsection the PhaseI algorithm on a simple example, namely on the transaction sequence $T = (\{a,b,c\}, \{a,b\}, \{b,d\})$ and the support sequence $\sigma = (0.3, 0.5, 0.4)$ (see Figure 4 below).

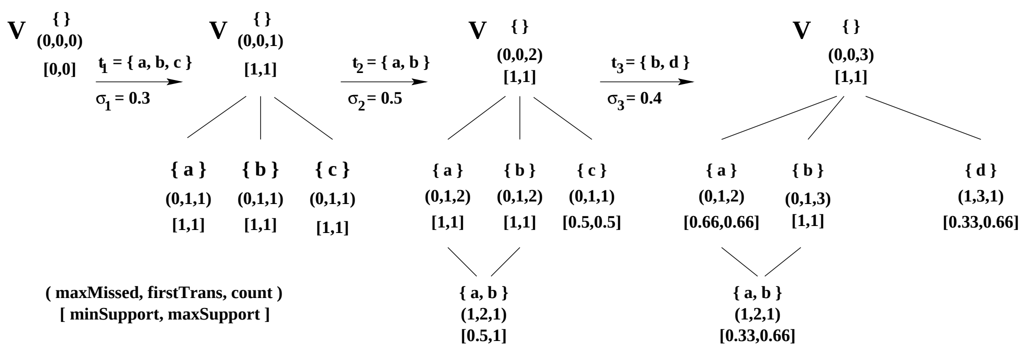

Figure 4

As indicated in figure 4, we denote by the triple the three associated integers for each set in the support lattice V and by the interval the bounds on its support.

We initialize V to $\{\emptyset\}$ and the associated integers of $\emptyset$ to $(0, 0, 0)$.

Reading $t_1 = \{a, b, c\}$ we first increment the count of $\emptyset$, since $\emptyset \subseteq t_1$. Because the empty set is the only strict subset of a singleton set and $1 = maxSupport(\emptyset) \geq \sigma_1$, we add all singleton subsets of t_1 to V. By equality (1) in Subsection 5.2 we get $maxMissed(v) = 0$ for each singleton set v. Hence we set the associated integers of v to $(0, 1, 1)$. Since there is no set in V with $maxSupport < 1$, we can not prune an itemset from V and the first transaction is processed.

Reading $t_2 = \{a, b\}$ we increment the count of $\emptyset$, $\{a\}$ and $\{b\}$ but not of $\{c\}$, since $\{c\} \not\subseteq t_2$. Since $\{a, b\} \subseteq t_2$ and all its subsets are in V with $maxSupport \geq \sigma_2 = 0.5$, we insert $\{a, b\}$ in V. Since $\lceil\sigma\rceil_1 = (0.3, 0, 0, \ldots)$ we get

$$\max\{\, \lfloor (2-1) avg_1(\lceil\sigma\rceil_1) \rfloor,\, 2 - 1 \,\} = 1.$$

Hence $maxMissed(\{a, b\}) = 1$ by equality (1), since $maxMissed(w) + count(w) = 2$ for all $w \subset \{a, b\}$. We set the associated integers to $(1, 2, 1)$. We note that $maxSupport(\{a, b\}) = 1$ is a sharp upper bound, since $support_2(\{a, b\}) = 1$. Since all $v \in V$ have support $maxSupport(v) \geq \sigma_2 = 0.5$, no set can be removed from V. The support interval for $\{c\}$ dropped from $[1, 1]$ to $[0.5, 0.5]$, since $\{c\}$ was not contained in t_2.

Reading $t_3 = \{b, d\}$ we increment the count of $\emptyset$ and $\{b\}$. Since the singleton $\{d\} \subseteq t_3$ is not contained in V, we include it as above. By $\lceil\sigma\rceil_2 = (0.5, 0.5, 0, 0, \ldots)$ we get $avg_2(\lceil\sigma\rceil_2) = 0.5$ and hence

$$\max\{\, \lfloor (3-1) \cdot 0.5 \rfloor,\, 1 - 1 \,\} = 1.$$

Since $maxMissed(\emptyset) + count(\emptyset) - 1 = 2 > 1$ we initialize the associated integers to $(1, 3, 1)$. Because $maxSupport(\{c\}) = 0.33 < 0.4 = \sigma_3$ we are free to remove $\{c\}$ from V and choose to do so. Since for all sets remaining in V the inequality $maxMissed \geq \sigma_3$ holds, there is no further set to prune. Hence the transaction t_3 is processed as well.

5.4 Support Thresholds

In this section we discuss the threshold given by Theorem 1, i.e. the support threshold for which PhaseI guarantees to include all large itemsets.

Let V be the support lattice constructed by PhaseI after the i-th transaction on a given transaction sequence and support sequence $\sigma = (\sigma_1, \sigma_2, \ldots)$. By Theorem 1 the support lattice V is a superset of all large itemsets relative to the the support threshold

$$\rho_i = \max\{\, avg_i(\lceil\sigma\rceil_i),\, \frac{c+1}{i}\,\} \tag{2}$$

where c is the maximal cardinality of a large itemset. If the user changed the support threshold, then ρ_i can be greater than σ_i, the current user-specified threshold. In this subsection, we discuss the relationship between ρ_i and σ_i. In particular:

1. $\rho_i = s$ for a constant support threshold s after the first $\frac{1}{s}(c+1)$ transactions.
2. The term $\frac{c+1}{i}$ is desirable.
3. The term $avg_i(\lceil\sigma\rceil_i)$ is a sharp lower bound relative to which V is a support lattice.
4. $\rho_i \to \sigma_i$ for $i \to \infty$ in typical scenarios.
5. Achieving $\rho_i = \sigma_i$ for changing support thresholds.

1) $\rho_i = s$ for a constant support threshold s after the first $\frac{1}{s}(c+1)$ transactions: Let σ be a constant support sequence, i.e. $\sigma = (s, s, \ldots)$. Thus the ceiling $\lceil\sigma\rceil_i$ is up to i also constant s, yielding $avg_i(\lceil\sigma\rceil_i) = s$. We now have $\rho_i = \max\{s,\, \frac{c+1}{i}\}$ and for $i > (c+1)/s$ we get $\rho_i = s$. Thus V is a superset of all large itemsets after the first $\frac{1}{s}(c+1)$ transactions.

2) The term $\frac{c+1}{i}$ is desirable: Suppose $\rho_i = avg_i(\lceil\sigma\rceil_i)$ would hold instead of (2). Hence we get $\rho_1 = \sigma_1$ for the first transaction. Every subset of t_1 must be contained in V after the first transaction t_1 is processed, since every subset of t_1 has support 1 in the initial transaction sequence (t_1). If t_1 consist of 30 items, V must contain all 2^{30} subsets, which is clearly not desirable. Thus the term $\frac{c+1}{i}$ protects the support lattice constructed by $PhaseI$ from an exponential blow-up during the first few transactions processed.

3) The term $avg_i(\lceil\sigma\rceil_i)$ is a sharp lower bound relative to which V is a support lattice. Let σ be an arbitrary support sequence. Since $\lceil\sigma\rceil_i$ is greater or equal to σ_i up to i, we get $avg_i(\lceil\sigma\rceil_i) \geq \sigma_i$ and hence $\rho_i \geq \sigma_i$. While V is a superset of all large itemsets relative to the support threshold ρ_i it is not guaranteed to be a superset with respect to σ_i. By the following example we show that ρ_i is a sharp lower bound on the support threshold for which V is a support lattice. Let $T = (t_1, \ldots, t_{100})$ and $\sigma = (\sigma_1, \ldots, \sigma_{100})$ with

$$t_j = \begin{cases} \{a\} & \text{for } j = 1, \ldots, 25 \\ \{b\} & \text{for } j = 26, \ldots, 100 \end{cases} \quad \text{and} \quad \sigma_j = \begin{cases} 0.1 & \text{for } j = 1, \ldots, 30 \\ 0.5 & \text{for } j = 31, \ldots, 51 \\ 0.0 & \text{for } j = 52, \ldots, 100 \end{cases} .$$

Hence

$$avg_{100}(\lceil\sigma\rceil_{100}) = 0.25 + \epsilon > 0.0 = \sigma_{100}$$

with $\epsilon = 0.005$. Since $support_{51}(\{a\}) < 0.5 = \sigma_{51}$ the algorithm is free to remove $\{a\}$ from V while processing transaction 51. Suppose $\{a\}$ is removed from V. Since $\{a\}$ is not contained in any transaction after dropping $\{a\}$ from V, the algorithm gets no indication to reinclude $\{a\}$ in V. Hence $\{a\} \notin V$ and therefore

$$avg_{100}(\lceil\sigma\rceil_{100}) = 0.25 + \epsilon > 0.25 = support_{100}(\{a\})$$

is a sharp lower lower (by adapting the above example we can make ϵ arbitrarily small).

4) $\rho_i \to \sigma_i$ for $i \to \infty$ in typical scenarios: We envision as a typical scenario, that the user initially changes the support threshold often, in reaction to the continously generated association rules. After the user has found a satisfactory threshold we do not expect further changes in the support threshold, i.e. $\sigma_j = \sigma_{j+1} = \ldots$ for some transaction index j. Hence

$$\rho_i \to \sigma_i \quad \text{for } i \to \infty.$$

5) Achieving $\rho_i = \sigma_i$ for changing support thresholds: Instead of the user-specified support sequence σ, we pass to PhaseI a sequence $\tilde{\sigma}$ which is pointwise less or equal to σ. We define $\tilde{\sigma}$ as follows: Let $\tilde{\sigma}_1 = \sigma_1$ and for $i > 1$ determine $\tilde{\sigma}_i$ by solving the equation

$$\max\{\, avg_i(\lceil\tilde{\sigma}\rceil_i), \frac{c+1}{i} \,\} = \sigma_i. \tag{3}$$

Since ρ_i then equals $\max\{\, avg_i(\lceil\tilde{\sigma}\rceil_i), \frac{c+1}{i} \,\}$ by Theorem 1 we get $\rho_i = \sigma_i$ by equality (3). We note that the solution of (3) can be negative, e.g. because $\sigma_i \ll \sigma_{i-1}$. In this case we define $\tilde{\sigma}_i$ by some small constant $\epsilon > 0$. If $\sigma_j \geq \delta > \epsilon$ for some $\delta \in [0, 1]$ then a non-negative solution for (3) exists after a finite number of transactions, yielding $\rho_i = \sigma_i$.

5.5 Pruning Strategy

PhaseI does not prescribe a particular pruning strategy as long as all supersets of a pruned itemset are pruned as well. Hence besides a drop-all (remove all small itemsets in V) or a no-drop strategy (do not remove any itemset from V) a so called negative-border strategy is also admissible, c.f. [Toi96, TBAR97].

PhaseI does not require that the support lattice is pruned after every transaction processed. Since pruning can be an expensive operation, we can prune the support lattice either in fixed intervals or decide while processing when to prune. Of particular interest are pruning strategies which facilitate PhaseII forward-pruning techniques. We introduce them in the following Section 6.

6 PhaseII Algorithm

In this section, we fully describe the PhaseII algorithm for the second scan. PhaseII computes the precise support of all large itemsets in the previously computed support lattice and continuously removes all small itemsets. In Subsection 6.1, we describe a preliminary PhaseII algorithm without forward pruning. In Subsection 6.2, we describe our forward-pruning technique. In Subsection 6.3, we give the formal definition of PhaseII, combining the preliminary PhaseII algorithm with the forward pruning technique.

6.1 Overview

Let V be the support lattice computed by PhaseI. PhaseII uses the last user-specified support threshold σ_n as pruning threshold.

Initially PhaseII removes all trivially small itemsets, i.e. itemsets with $maxSupport < \sigma_n$, from V.

Scanning the transaction sequence PhaseII increments *count* and decrements $maxMissed$ for each itemset contained in the current transaction, up to the transaction at which the itemset was inserted. Setting $maxMissed$ to 0 we get $minSupport = maxSupport$, the actual support of the itemset. We remove the itemset if it is small. Setting $maxMissed(v) = 0$ for an itemset v may yield $maxSupport(w) > maxSupport(v)$ for some superset w of v.

Thus we set $maxMissed(w) = count(v) - count(w)$ for all supersets w of v with $maxSupport(w) > maxSupport(v)$.

PhaseII stops as soon as the current transaction index is past $firstTrans$ for all itemsets in the lattice. The resulting lattice contains all large itemsets along with the precise support for each itemset.

6.2 Forward Pruning

We extend the preliminary PhaseII algorithm described above by a "forward pruning" technique, which allows us to remove some small singleton set v and all its descendants from V, before we reach $firstTrans(v)$ even if $maxSupport(v) \geq \sigma_n$. The general idea is the following:

Let v be a singleton set in V and suppose $support_n(v) \geq \sigma_n$. Suppose we are rescanning the i-th transaction. For a real number x we denote by $\lceil x \rceil$ the least integer greater or equal to x, i.e. $\lceil x \rceil = \min\{i \in \mathbb{Z} \mid x \leq i\}$. Thus there are at least $\lceil n \cdot \sigma_n \rceil - count(v)$ occurences of v in $t_{i+1}, \ldots, t_{ft-1}$ with $ft = firstTrans(v)$. Suppose that v was not contained in V after the i-th transaction was processed by PhaseI. At the first occurence of v after t_i, PhaseI inserts v in V with $maxMissed(v) \geq \lfloor i \cdot avg_i(\lceil \sigma \rceil_i) \rfloor$, since v is a singleton. The insertion of v in V is guaranteed to take place, since its only subset is the emptyset which always has support 1. By an induction on $ft - i$ we get that if

$$\lceil n \cdot \sigma \rceil - count(v) + \lfloor i \cdot avg_i(\lceil \sigma \rceil_i) \rfloor > \lfloor (ft-1) avg_{ft-1}(\lceil \sigma \rceil_{ft-1}) \rfloor \qquad (4)$$

then v had to be in V while the ft-th transaction was processed by PhaseI, c.f. Lemma 8. This is in contradiction to the insertion of v in V by PhaseI during the ft-th transaction. Hence $support_n(v) < \sigma_n$ and we prune v and all its descendents from V while PhaseII processes the i-th transaction. Note that this arguments requires that $v \notin V$ at the i-th transaction in PhaseI and that inequality (4) holds. The following Theorem 2 asserts the correctness of our "forward pruning" technique:

Theorem 2 *Let T be a transaction sequence of length n, $\sigma = (\sigma_1, \ldots, \sigma_n)$ a support sequence and V the support lattice returned by PhaseI(T, σ). Let $ft = firstTrans_n(v)$ and i some index $< ft$. If v is a singelton set which* *does not occur in the first i transactions and*

$$\lceil n \cdot \sigma_n \rceil - count(v) + \lfloor i \cdot avg_i(\lceil \sigma \rceil_i) \rfloor > \lfloor (ft-1) avg_{ft-1}(\lceil \sigma \rceil_{ft-1}) \rfloor$$

then

$$support_n(v) < \sigma_n.$$

Proof: see Theorem 5 in Appenix B. $\square$.

For a straight forward generalization of Theorem 2 to a set v of arbitrary cardinality we need to know: *1)* that $v \notin V$ at the i-th transaction of PhaseI and *2)* that PhaseI inserts v in the support lattice before transaction ft if the inequality holds. However, for non-singleton sets, this requires knowledge about the PhaseI pruning strategy. Since we do not prescribe a particular pruning strategy, this knowledge is, in general, not available.

6.3 PhaseII Algorithm

Adding the forward pruning technique to the preliminary PhaseII algorithm described in Subsection 6.1 we get our PhaseII algorithm:

```
Function PhaseII( support lattice V, transaction sequence (t_1, ..., t_n),
                  support sequence σ ) : support lattice;
integer ft, i = 0;
begin
 InitialPrune:  V := V\{v ∈ V | maxSupport(v) < σ_n };
 Rescan:        while ∃v ∈ V : i < firstTrans(v) do
                  i++;
                  for all v ∈ V do
                    ft := firstTrans(v);
 Adjust:            if v ⊆ t_i and ft < i then
                      count(v)++, maxMissed(v)--;
                    fi;
                    if ft = i then
                      maxMissed(v) := 0;
                      for all w ∈ V : v ⊂ w and
                                 maxSupport(w) > maxSupport(v) do
                        maxSupport(w) := count(v) − count(w);
                    od; fi;
 Prune:             if maxSupport(v) < σ_n then V := V\{v}; fi;
 Forward
   Prune:           if |v| = 1 and v does not occur in t_1, ..., t_i and
                      ⌈n · σ_n⌉ − count(v) + ⌊i · avg_i(⌈σ⌉_i)⌋
                      > ⌊(ft − 1)avg_{ft−1}(⌈σ⌉_{ft−1})⌋ then
                        V := V\{ w ∈ V | v ⊆ w };
                fi; od; od;
                return V;
end;
```

Figure 5

7 CARMA

Combining PhaseI with PhaseII we get our CARMA algorithm:

Function CARMA(transaction sequence T, support sequence σ) :
support lattice;
support lattice V;
begin
  V := PhaseI(T, σ);
  V := PhaseII(V, T, σ);
  return V;
end;

Figure 6

We note that although we formally pass the support sequence σ as a parameter to PhaseI, σ is actually determined by the user while PhaseI scans the transaction sequence. The generation of the association rules from the support lattice and the displaying of them to the user should also be placed in PhaseI and PhaseII.

An implementation of CARMA is under way.

In the following theorem we state the correctness of CARMA:

Theorem 3 *Let T be a transaction sequence of length n, σ a support sequence, V the support lattice computed by CARMA(T, σ) and c the maximal cardinality of an itemset in V. Then V is a support lattice relative to the threshold*

$$\max\{\, avg_n(\lceil\sigma\rceil_n),\ \tfrac{c+1}{n} \,\}.$$

For an itemset v we have in particular:

- *If $v \subseteq I$ satisfies $support_n(v) > \max\{\, avg_n(\lceil\sigma\rceil_n),\ \frac{|v|-1}{n} \,\}$ then $v \in V$.*
- *If $v \in V$ then $support_n(v) \geq \sigma_n$ and*
$$minSupport(v) = support_n(v) = maxSupport(v).$$
- *If σ is constant s and $n \geq \frac{1}{s}(c+1)$ then V is a support lattice for the support threshold s.*

Proof: By Theorem 1, Subsection 6.1 and Theorem 2. $\square$

A Proof of Correctness for PhaseI

In this section we give a proof for the correctness of the PhaseI algorithm.

For convenience, we introduce the following conventions: Let I and K be some sets. By $K \subseteq I$ we denote set inclusion and by $K \subset I$ strict set inclusion, i.e. $K \subset I$ if and only if $K \subseteq I$ and $K \neq I$. By $I \backslash K$ we denote set exclusion, i.e. the set $\{x \in I \mid x \notin K\}$. By $\mathbb{N}$ we denote the natural numbers including 0 and by $\mathbb{Z}$ the integers. For a real number x we denote by $\lceil x \rceil$ the least integer greater or equal to x, i.e. $\lceil x \rceil = \min\{i \in \mathbb{Z} \mid x \leq i\}$ and by $\lfloor x \rfloor$ the largest integer less or equal to x, i.e. $\lfloor x \rfloor = \max\{i \in \mathbb{Z} \mid x \geq i\}$. Let V_i for some $i \in \mathbb{N}$ be a sublattice of the subset lattice of I. For $t \subseteq I$ let

$$subsets_i(t) := \{v \in V_i \mid v \subset t\} \quad and \quad supersets_i(t) := \{v \in V_i \mid t \subset v\}.$$

Note that t itself is neither contained in $subsets_i(t)$ nor in $supersets_i(t)$.

To facilitate the proof we state in the following definition the PhaseI algorithm in its recursive form:

Definition 1 *Let $T = (t_1, t_2, \ldots)$ be a transaction sequence relative to some set I and let $\sigma = (\sigma_1, \sigma_2, \ldots)$ be a support sequence. We define a sublattice V_i of the subset lattice of I and functions $maxMissed_i$, $firstTrans_i : V_i \to \mathbb{N}$ and $count_i$ by induction on i:*

Let $i = 0$: Let $V_0 = \{\emptyset\}$, $maxMissed_0(\emptyset) = 0$, $firstTrans_0(\emptyset) = 0$ and $count_0(\emptyset) = 0$.

Let $i > 0$ and suppose we have defined V_j for all $j < i$:

For all $v \in V_{i-1}$ let

$$maxMissed_i(v) = maxMissed_{i-1}(v),\ firstTrans_i(v) = firstTrans_{i-1}(v)$$

and

$$count_i(v) = \begin{cases} count_{i-1}(v) & \text{if } v \not\subseteq t_i \\ count_{i-1}(v) + 1 & \text{if } v \subseteq t_i \end{cases}.$$

Let

$$C_i = \{\, v \subseteq t_i \mid v \notin V_{i-1} \text{ and } \forall w \subset v : w \in V_{i-1}, maxSupport_i(w) \geq \sigma_i \,\}$$

and

$$D_i \subseteq \{v \in V_{i-1} \mid maxSupport_i(v) < \sigma_i\}$$

such that if $v \in D_i$ then $supersets_{i-1}(v) \subseteq D_i$.

Let $V_i = (V_{i-1} \cup C_i) \backslash D_i$. For $v \in V_i \backslash V_{i-1}$ let

$$\begin{aligned} maxMissed_i(v) = \min\{ \quad & \max\{\, \lfloor (i-1) \cdot avg_{i-1}(\lceil \sigma \rceil_{i-1}) \rfloor, |v| - 1 \,\}, \\ & maxMissed_{i-1}(w) + count_{i-1}(w) \mid \\ & w \in subsets_{i-1}(v) \,\}, \end{aligned}$$

$count_i(v) = 1$ *and* $firstTrans_i(v) = i$.

First, we assert the correctness of our recursive definition of PhaseI:

Lemma 1 *Let T be a transaction sequence of length n and σ a support sequence. Let V be the lattice computed by $PhaseI(T, \sigma)$ and define V_n as in Definition 1. Define D_i in Definition 1 according to the pruning strategy chosen for $PhaseI$.*

Then $V = V_n$ and for each $v \in V$ the corresponding associated integers are equal, i.e. $maxMissed(v) = maxMissed_n(v)$, $firstTrans(v) = firstTrans_n(v)$ and $count(v) = count_n(v)$.

Proof: By induction on n. □

By the following lemma we can easily compute ceilings of a support sequence.

Lemma 2 *Let $\sigma = (\sigma_1, \sigma_2, \ldots)$ be a support sequence. Then*

$$\lceil\sigma\rceil_{1,1} = \sigma_1 \quad and \quad \lceil\sigma\rceil_{1,j} = 0 \;\; for\ j \geq 2.$$

and for $i > 1$ we have

$$\lceil\sigma\rceil_{i,j} = \begin{cases} \lceil\sigma\rceil_{i-1,j} & for\ j < i\ and\ \lceil\sigma\rceil_{i-1,j} > \sigma_i \\ \sigma_i & for\ j \leq i\ and\ \lceil\sigma\rceil_{i-1,j} \leq \sigma_i \\ 0 & for\ j > i \end{cases}.$$

Proof: By induction on i and the definition of support ceilings. □

We summarize some observations on ceilings:

Lemma 3 *Let $\sigma = (\sigma_1, \sigma_2, \ldots)$ be a support sequence and i a positive integer. Then*

1. $\lceil\sigma\rceil_{i+1,j} \geq \lceil\sigma\rceil_{i,j}$ *for all j,*
2. $avg_j(\lceil\sigma\rceil_i) \geq \sigma_j$ *for all $j \leq i$,*
3. $avg_j(\lceil\sigma\rceil_i) \geq avg_j(\lceil\sigma\rceil_j)$ *for all $j \leq i$,*

Proof: By Lemma 2. $\square$

For a transaction sequence T and an itemset v we denote by $countT_i(v)$ the number of occurences of v in the first i transactions of T.

Lemma 4 *Let T be a transaction sequence, σ a support sequence and i an integer. Define V_i relative to T and σ as in Definition 1. Let $v \in V_i$. Then*

$$count_i(v) = countT_i(v) - countT_{firstTrans_i(v)-1}(v)$$

and $v \subseteq t_{firstTrans_i(v)}$.

Proof: By induction on i. $\square$

Lemma 5 *Let T be a transaction sequence, σ a support sequence and i an integer. Define V_i relative to T and σ as in Definition 1. Let $v, w \in V_i$ and $w \subseteq v$. Then*

$$\begin{aligned} maxMissed_i(w) &\leq maxMissed_i(v), \\ firstTrans_i(w) &\leq firstTrans_i(v), \\ count_i(w) &\geq count_i(v), \\ maxMissed_i(w) + count_i(w) &\geq maxMissed_i(v) + count_i(v). \end{aligned}$$

Proof: By Definition 1 and induction on i. $\square$

Lemma 6 *Let T be a transaction sequence, σ a support sequence and i an integer. Define V_i relative to T and σ as in Definition 1. Suppose $v \in V_i$ and $w \subseteq v$. Then*

$$w \in V_i.$$

Proof: Let $T = (t_1, t_2, \ldots)$ be a transaction sequence and $\sigma = (\sigma_1, \sigma_2, \ldots)$ a support sequence. Let $v \in V_i$ and $w \subseteq v$. We may assume, without loss of generality, that $w \subset v$. We prove Lemma 6 by induction on i.

Let $i = 0$. Hence $v = \emptyset = w$. Thus Lemma 6 holds trivially.

Let $i \geq 1$ and suppose Lemma 6 holds for all V_j with $j < i$.

1) Suppose $v \in V_{i-1}$. Hence $w \in V_{i-1}$ by the induction hypothesis. Since $v \in V_i$ we have $v \notin D_i$. Thus w is also not in D_i because otherwise $v \in supersets_{i-1}(w) \subseteq D_i$, in contradiction to $v \in V_i$. Therefore $w \in V_i$.

2) Suppose $v \notin V_{i-1}$ and $w \in V_{i-1}$. By $v \in V_i$ we have $v \subseteq t_i$ and $v \in C_i$. Since $w \subset v$ we have $w \in subsets_{i-1}(v)$ and therefore $maxSupport_i(w) \geq \sigma_i$ since $v \in C_i$. Thus $w \notin D_i$ and since $w \in V_{i-1}$ we have $w \in V_i$.

3) Suppose $v \notin V_{i-1}$ and $w \notin V_{i-1}$. By $v \in V_i$ we have $v \subseteq t_i$ and $v \in C_i$. Since $w \subset v$ we get $w \in V_{i-1}$ by Definition 1, in contradiction to $w \notin V_{i-1}$. Hence this case does not occur. □

Lemma 7 *Let T be a transaction sequence, σ a support sequence and i an integer. Define V_i relative to T and σ as in Definition 1. Then $\emptyset \in V_i$, $maxMissed_i(\emptyset) = 0$, $firstTrans_i(\emptyset) = 0$ and $count_i(\emptyset) = i$.*

Proof: By induction on i since $\emptyset$ is a subset of all transactions in T. □

Proposition 1 *Let T be a transaction sequence relative to some set I, σ a support sequence and V_i a support lattice relative to T and σ up to some positive integer i. Let v be a subset of I.*

1. *If $v \in V_i$ then $countT_{firstTrans_i(v)-1}(v) \leq maxMissed_i(v)$*

2. *If $support_i(v) > \max\{\, avg_i(\lceil\sigma\rceil_i),\, \frac{|v|-1}{i} \,\}$ then $v \in V_i$.*

Proof: Let $T = (t_1, t_2, \ldots)$ be a transaction sequence relative to some set I, $\sigma = (\sigma_1, \sigma_2, \ldots)$ a support sequence and v a subset of I. We prove Proposition 1 by double induction on $c = |v|$ and on i:

Let $c = 0$. Hence $v = \emptyset$. Thus Proposition 1 holds for all i by Lemma 7.

Let $c \geq 1$. Suppose Proposition 1 holds for all subsets of I with less than c elements and all i. Let $v \subseteq I$ such that $c = |v|$. We show by induction on i that *1.* and *2.* hold.

Let $i = 1$.

1. Let $v \in V_1$. By Definition 1 we have $V_1 = \{\emptyset\} \cup \bigcup_{a \in t_1} \{a\}$ and

$$maxMissed_1(v) = 0 = countT_0(v)$$

for all $v \in V_1$. Hence *1.* holds for $i = 1$.

2. Let $support_1(v) > \max\{\, avg_1(\lceil\sigma\rceil_1),\, \frac{|v|-1}{1} \,\}$. Since $c \geq 1$ we have

$$1 \geq support_1(v) > \max\{\, avg_1(\lceil\sigma\rceil_1),\, c-1 \,\} \geq 0.$$

Hence $c = 1$. Since $support_1(v) > 0$ we get $v \subseteq t_1$. Hence $v \in V_1 = \{\emptyset\} \cup \bigcup_{a \in t_1} \{a\}$.

Let $i > 1$. Suppose *1.* and *2.* hold for all i if $v \subseteq I$ contains less than c elements and up to $i - 1$ if v contains c elements. We show that *1.* and *2.* hold for i as well if v contains c elements:

1. Let $v \in V_i$ and $|v| = c$.

i) Suppose $v \in V_{i-1}$. Thus we get by the induction hypothesis and Definition 1

$$countT_{firstTrans_i(v)-1}(v) \leq maxMissed_{i-1}(v) = maxMissed_i(v).$$

ii) Suppose $v \notin V_{i-1}$ and there exists a set $w \in subsets_{i-1}(v)$ such that $maxMissed_i(v) = maxMissed_{i-1}(w) + count_{i-1}(w)$. Since $w \subset v$ we have by the induction hypothesis for *1.* and Lemma 4

$$countT_{i-1}(v) \leq countT_{i-1}(w) \leq maxMissed_{i-1}(w) + count_{i-1}(w).$$

Since $v \notin V_{i-1}$ but $v \in V_i$ we have $v \subseteq t_i$. By Definition 1 we therefore get

$$\begin{aligned} countT_i(v) &= countT_{i-1}(v) + 1 \leq countT_{i-1}(w) + 1 \\ &\leq maxMissed_{i-1}(w) + count_{i-1}(w) + 1 \\ &= maxMissed_i(v) + 1 \\ &= maxMissed_i(v) + count_i(v). \end{aligned}$$

Hence *1.* follows by Lemma 4 for this case.

iii) Suppose $v \notin V_{i-1}$ and no vertex $w \in subsets_{i-1}(v)$ exists such that $maxMissed_i(v) = maxMissed_{i-1}(w) + count_{i-1}(w)$. Hence $v \in C_i$,

$$maxMissed_i(v) = \max\{\, \lfloor (i-1) \cdot avg_{i-1}(\lceil \sigma \rceil_{i-1}) \rfloor,\, c-1 \,\},$$

and $firstTrans_i(v) = i$. Suppose $countT_{i-1}(v) > maxMissed_i(v)$. Since $countT_{i-1}(v) \in \mathbb{N}$ we get by the equality above

$$support_{i-1}(v) > \max\{\, avg_{i-1}(\lceil \sigma \rceil_{i-1}),\, \tfrac{c-1}{i} \,\}.$$

Thus $v \in V_{i-1}$ by the induction hypothesis for *2.*, in contradiction to $v \notin V_{i-1}$. Hence $countT_{i-1}(v) \leq maxMissed_i(v)$.

2. Let $v \subseteq I$ such that $|v| = c$ and $support_i(v) > \max\{\, avg_i(\lceil \sigma \rceil_i),\, \frac{c-1}{i} \,\}$.

i) Suppose $v \not\subseteq t_i$. By Lemma 3 we get

$$\begin{aligned} countT_{i-1}(v) &= countT_i(v) = i \cdot support_i(v) \\ &> \max\{\, i \cdot avg_i(\lceil \sigma \rceil_i),\, c-1 \,\} = \max\{\, \sum_{j=1}^{i} \lceil \sigma \rceil_{i,j},\, c-1 \,\} \\ &\geq \max\{\, \sum_{j=1}^{i-1} \lceil \sigma \rceil_{i-1,j},\, c-1 \,\} \\ &= \max\{\, (i-1) \cdot avg_{i-1}(\lceil \sigma \rceil_{i-1}),\, c-1 \,\}. \end{aligned}$$

Hence $support_{i-1}(v) > \max\{\, avg_{i-1}(\lceil \sigma \rceil_{i-1}),\, \frac{c-1}{i-1} \,\}$. By the induction hypothesis for *2.* we get $v \in V_{i-1}$. By the induction hypothesis for *1.* and by $v \not\subseteq t_i$

we get

$$\begin{aligned} maxMissed_{i-1}(v) + count_{i-1}(v) &\geq countT_{i-1}(v) = countT_i(v) \\ &> \max\{\, i \cdot avg_i(\lceil\sigma\rceil_i),\, c-1\,\}. \end{aligned}$$

Also by $v \not\subseteq t_i$ we have
$maxMissed_i(v) + count_i(v) = maxMissed_{i-1}(v) + count_{i-1}(v)$.
Hence

$$maxMissed_i(v) + count_i(v) > \max\{\, i \cdot avg_i(\lceil\sigma\rceil_i),\, c-1\,\}. \tag{5}$$

Suppose $v \in D_i$. Hence $i \cdot \sigma_i > maxMissed_i(v) + count_i(v)$ and therefore
$i \cdot avg_i(\lceil\sigma\rceil_i) > maxMissed_i(v) + count_i(v)$
by Lemma 3, in contradiction to inequality (5). Hence $v \notin D_i$ and therefore $v \in V_i$.

ii) Suppose $v \subseteq t_i$ and $v \in V_{i-1}$. Since $v \subseteq t_i$ we have $count_i(v) = count_{i-1}(v) + 1$. Since $v \in V_{i-1}$ we get by the induction hypothesis for *1.* and Lemma 4

$$\begin{aligned} maxMissed_i(v) + count_i(v) &= maxMissed_{i-1}(v) + count_{i-1}(v) + 1 \\ &\geq countT_{i-1}(v) + 1 = countT_i(v) \\ &> i \cdot avg_i(\lceil\sigma\rceil_i) > i \cdot \sigma_i \end{aligned}$$

Hence $maxSupport_i(v) > \sigma_i$. If $v \in D_i$ then $maxSupport_i(v) < \sigma_i$, a contradiction. Thus $v \notin D_i$ and $v \in V_i$.

iii) Suppose $v \subseteq t_i$ and $v \notin V_{i-1}$. Let w be a subset of v of cardinality $c-1$. By $w \subset v \subseteq t_i$ we have

$$\begin{aligned} countT_{i-1}(w) + 1 &= countT_i(w) \geq countT_i(v) \\ &> \max\{\, i \cdot avg_i(\lceil\sigma\rceil_i),\, c-1\,\} \\ &= \max\{\, \sum_{j=1}^{i} \lceil\sigma\rceil_{i,j},\, c-1\,\} \\ &\geq \max\{\, \sum_{j=1}^{i-1} \lceil\sigma\rceil_{i-1,j},\, c-1\,\} \\ &= \max\{\, (i-1) \cdot avg_{i-1}(\lceil\sigma\rceil_{i-1}),\, c-1\,\} \end{aligned}$$

Thus $support_{i-1}(w) > \max\{\, avg_{i-1}(\lceil\sigma\rceil_{i-1}),\ \frac{c-2}{i-1}\,\}$ and therefore $w \in V_{i-1}$ by the induction hypothesis for *2.* By Lemma 6 all subsets $u \subset v$ are therefore elements of V_{i-1}. By the induction hypothesis for *1.* we get

$$\begin{aligned} maxMissed_i(u) + count_i(u) &\geq countT_i(u) \geq countT_i(v) \\ &> i \cdot avg_i(\lceil\sigma\rceil_i) \geq i \cdot \sigma_i \end{aligned}$$

for all $u \subset v$. Hence v is an element of C_i and therefore of V_i. □

Theorem 4 *Let T be a transaction sequence of length n, σ a support sequence and V the subset lattice computed by $PhaseI(T, \sigma)$. Then for any itemset v*

$$support_n(v) > \max\{\, avg_n(\lceil\sigma\rceil_n),\ \tfrac{|v|-1}{n}\,\} \quad \textit{implies} \quad v \in V.$$

Let $c = \max\{|v|$ for $v \in V$ with $maxSupport(v) \geq \sigma_n\}$, i.e. the maximal cardinality of all potentially large itemsets in V. Then V is a support lattice up to n relative to T and support threshold

$$\max\{\, avg_n(\lceil\sigma\rceil_n),\ \tfrac{c+1}{n}\,\}.$$

Proof: Let $T = (t_1, \ldots, t_n)$ be a transaction sequence relative to some set I, $\sigma = (\sigma_1, \ldots, \sigma_n)$ a support sequence and let V_n be the lattice defined by Definition 1 with D_i defined according to the pruning strategy chosen for $PhaseI$. By Lemma 1 it suffices to proof Theorem 4 for V_n. Let v be a subset of I. By Proposition 1 we have $v \in V_n$ if

$$support_n(v) > \max\{\, avg_n(\lceil\sigma\rceil_n),\ \frac{|v|-1}{n}\,\}. \qquad (6)$$

By Lemma 4 and Proposition 1 we get that V_n is a support lattice up to n relative to T. Let $c = \max\{|v| : v \in V, maxSupport(v) \geq \sigma_n\}$. By Lemma 7 we may assume, without loss of generality, that $c \geq 1$. We show that V_n is a support lattice relative to the support threshold

$$\max\{\, avg_n(\lceil\sigma\rceil_n),\ \tfrac{c+1}{n}\,\}.$$

Let v be a subset of I such that $support_n(v) \geq \max\{\, avg_n(\lceil\sigma\rceil_n),\ \frac{c+1}{n}\,\}$. Suppose $|v| \geq c+1$. Thus v contains a subset w of cardinality $c+1$. Since $support_n(w) \geq support_n(v)$ inequality (6) holds for w and thereby $w \in V_n$, in contradiction to the definition of c. Hence $|v| \leq c$. Since inequality (6) holds for this case we get $v \in V_n$. Hence V_n is a support lattice relative to the support threshold $\max\{\, avg_n(\lceil\sigma\rceil_n),\ \frac{c+1}{n}\,\}$. □

B Proof of Correctness for PhaseII

Lemma 8 *Let T be a transaction sequence, σ a support sequence and V_j for $j \geq 1$ a support lattice relative to T and σ up to j. Suppose the singelton itemset $\{a\}$ is not contained in V_i and*

$$countT_n(\{a\}) - countT_i(\{a\}) + \lfloor \sum_{k=1}^{i} \lceil \sigma \rceil_{i,k} \rfloor > \lfloor \sum_{k=1}^{n} \lceil \sigma \rceil_{n,k} \rfloor \qquad (7)$$

for some positive integers i and n with $i \leq n$. Then

$$\{a\} \in V_n.$$

Proof: We prove Lemma 8 by induction on $d = n - i$. Let $T = (t_1, t_2, \ldots)$ and $\sigma = (\sigma_1, \sigma_2, \ldots)$. Let $n \geq i$, $a \in I$ such that $\{a\} \notin V_i$ and suppose that inequality (7) holds. For $d = 0$ the inequality is never satisfied and thus Lemma 8 holds trivially.

Let $d = 1$. Hence $n = i + 1$. Since $\sum_{k=1}^{n} \lceil \sigma \rceil_{n,k} \geq \sum_{k=1}^{i} \lceil \sigma \rceil_{i,k}$ we get by inequality (7)

$$countT_{i+1}(\{a\}) - countT_i(\{a\}) \geq 1.$$

Thus $a \in t_{i+1}$. By $\{a\} \notin V_i$ and Definition 1 we get $\{a\} \in V_{i+1} = V_n$. Hence Lemma 8 holds for this case.

Let $d > 1$ and suppose Lemma 8 holds if $n - i < d$.

Suppose $a \notin t_{i+1}$. Hence $countT_{i+1}(\{a\}) = countT_i(\{a\})$. Thus Lemma 8 holds for this case by the induction hypothesis.

Suppose $a \in t_{i+1}$. Since $\{a\} \notin V_i$ we have $\{a\} \in V_{i+1}$ and

$$\lfloor \sum_{k=1}^{i} \lceil \sigma \rceil_{i,k} \rfloor \leq maxMissed_{i+1}(\{a\}) \qquad (8)$$

by Definition 1. If $\{a\} \in V_j$ for all $j = i+2, \ldots, n$ then Lemma 8 holds trivially. Suppose that there exists an index j such that $\{a\} \notin V_j$. We may assume, without loss of generality, that j is minimal. Hence $\{a\} \in D_j$ with D_j defined as in Definition 1. Thus

$$count_j(\{a\}) + maxMissed_j(\{a\}) < j \cdot \sigma_j.$$

Since j is minimal we have $maxMissed_{i+1}(\{a\}) = maxMissed_j(\{a\})$. Together with the above inequality and (8) we have

$$count_j(\{a\}) + \lfloor \textstyle\sum_{k=1}^{i} \lceil \sigma \rceil_{i,k} \rfloor < j \cdot \sigma_j.$$

Hence we get by Lemma 3 and Lemma 4

$countT_j(\{a\}) - countT_i(\{a\}) + \lfloor\sum_{k=1}^{i} \lceil\sigma\rceil_{i,k}\rfloor < \lfloor j \cdot \sigma_j\rfloor \leq \lfloor\sum_{k=1}^{j} \lceil\sigma\rceil_{j,k}\rfloor$.
By (7) we now have

$$\begin{aligned} & countT_n(\{a\}) - countT_i(\{a\}) + \lfloor\sum_{k=1}^{i} \lceil\sigma\rceil_{i,k}\rfloor \\ - \quad & countT_j(\{a\}) + countT_i(\{a\}) - \lfloor\sum_{k=1}^{i} \lceil\sigma\rceil_{i,k}\rfloor \\ > \quad & \lfloor\sum_{k=1}^{n} \lceil\sigma\rceil_{n,k}\rfloor - \lfloor\sum_{k=1}^{j} \lceil\sigma\rceil_{j,k}\rfloor \end{aligned}$$

yielding $countT_n(\{a\}) - countT_j(\{a\}) + \lfloor\sum_{k=1}^{j} \lceil\sigma\rceil_{j,k}\rfloor > \lfloor\sum_{k=1}^{n} \lceil\sigma\rceil_{n,k}\rfloor$. Since $j > i$ we get $\{a\} \in V_n$ by the induction hypothesis. $\square$

Theorem 5 *Let T be a transaction sequence, σ a support sequence and V the support lattice returned by PhaseI(T, σ). Let $v \in V$ be a singleton set, i.e. $|v| = 1$, $ft = firstTrans_n(v)$ and $countT_i(v) = 0$ for some positive integer $i < ft$. If*

$$\lceil n \cdot \sigma_n\rceil - count(v) + \lfloor i \cdot avg_i(\lceil\sigma\rceil_i)\rfloor > \lfloor(ft-1)avg_{ft-1}(\lceil\sigma\rceil_{ft-1})\rfloor \tag{9}$$

then

$$support_n(v) < \sigma_n.$$

Proof: Let $\{a\} \in V_n$, $ft = firstTrans_n(\{a\}) > 1$ and $countT_i(\{a\}) = 0$. Suppose (9) holds and $support_n(\{a\}) \geq \sigma_n$. Hence $countT_n(\{a\}) \geq \lceil n \cdot \sigma_n\rceil$ and therefore $countT_{ft-1}(\{a\}) \geq \lceil n \cdot \sigma_n\rceil - count_n(\{a\})$ by Lemma 4. Since $countT_i(\{a\}) = 0$ we get by inequality (9)
$countT_{ft-1}(\{a\}) - countT_i(\{a\}) + \lfloor\sum_{k=1}^{i} \lceil\sigma\rceil_{i,k}\rfloor > \lfloor\sum_{k=1}^{ft-1} \lceil\sigma\rceil_{ft-1,k}\rfloor$.
Hence $\{a\}$ is an element of V_{ft-1} by Lemma 8. Since $firstTrans_n(\{a\}) = ft$ we have $\{a\} \in V_{ft} \backslash V_{ft-1}$, in contradiction to $\{a\} \in V_{ft-1}$. Thus

$$support_n(\{a\}) < \sigma_n.$$

$\square$

Acknowledgement: I would like to thank Joseph M. Hellerstein, UC Berkeley, for his inspiration, guidance and support. I am thankful to Ron Avnur for the many insightful discussions and to Retus Sgier, for his suggestions and support. Also, I would like to thank the Stanford database group for pointing out the applicability of CARMA to transaction sequences read from a network.

Christian Hidber
International Computer Science Institute
1947 Center Street, Suite 600
Berkeley, California 94704-1198

References

[AIS93] R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In *Proc. of the ACM SIGMOD Conference on Management of Data*, pages 207–216, Washington, D.C., May 1993.

[AS94] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proc. of the 20th Int'l Conf. on Very Large Databases*, Santiago, Chile, Sept. 1994.

[AY97] Charu C. Aggarwal and Philip S. Yu. Online generation of association rules. Technical Report RC 20899 (92609), IBM Research Division, T.J. Watson Research Center, Yorktown Heights, NY, June 1997.

[AY98] Charu C. Aggarwal and Philip S. Yu. Mining large itemsets for association rules. *Bulletin of the IEEE Computer Society Technical Comittee on Data Engineering*, pages 23–31, March 1998.

[BMUT97] Sergey Brin, Rajeev Motwani, Jeffrey D. Ullman, and Shalom Tsur. Dynamic itemset counting and implication rules for market basket data. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, volume 26,2 of *SIGMOD*

Record, pages 255–264, New York, May 13th–15th 1997. ACM Press.

[CHNW96] D. Cheung, J. Han, V. Ng, and C.Y. Wong. Maintenance of discovered association rules in large databases: An incremental updating technique. In *Proc. of 1996 Int'l Conf. on Data Engineering (ICDE'96)*, New Orleans, Louisiana, USA, Feb. 1996.

[CLK97] David W. L. Cheung, S.D. Lee, and Benjamin Kao. A general incremental technique for maintaining discovered association rules. In *Proceedings of the Fifth International Conference On Database Systems For Advanced Applications*, pages 185–194, Melbourne, Australia, March 1997.

[Hel96] Joseph M. Hellerstein. The case for online aggregation. Technical Report UCB//CSD-96-908, EECS Computer Science Divison, University of California at Berkeley, 1996.

[HHW97] Joseph M. Hellerstein, Peter J. Haas, and Helen J. Wang. Online aggregation. SIGMOD '97, 1997.

[SON95] A. Savasere, E. Omiecinski, and S. Navathe. An efficient algorithm for mining association rules in large databases. In *Proceedings of the Very Large Data Base Conference*, September 1995.

[TBAR97] Shiby Thomas, Sreenath Bodagala, Khaled Alsabti, and Sanjay Ranka. An efficient algorithm for the incremental updation of association rules in large databases. In *Proceedings of the 3rd International conference on Knowledge Discovery and Data Mining (KDD 97)*, New Port Beach, California, August 1997.

[Toi96] Hannu Toivonen. Sampling large databases for association rules. In T. M. Vijayaraman, Alejandro P. Buchmann, C. Mohan, and Nandlal L. Sarda, editors, *VLDB'96, Proceedings of 22th International Conference on Very Large Data Bases*, Mumbai (Bombay), India, September 1996. Morgan Kaufmann.

[ZPLO96] Mohammed Javeed Zaki, Srinivasan Parthasarathy, Wei Li, and Mitsunori Ogihara. Evaluation of sampling for data mining of association rules. Technical Report 617, Computer Science Dept., U. Rochester, May 1996.