Analysis of Candidate-Based Frequent Item-Set Algorithms

Nele Dexters[1]* Paul W. Purdom[2]
Dirk Van Gucht[2]*
nele.dexters@ua.ac.be, {pwp,vgucht}@cs.indiana.edu
[1] University of Antwerp
[2] Indiana University
* Contact author. Address: Middelheimlaan 1, 2020 Antwerpen, Belgium
tel.: (+32)3 265 38 69, fax: (+32)3 265 37 77

Abstract

We analyse several candidate-based frequent item-set algorithms, for the general shopping model where each combination of items has its own probability of being purchased, so any correlation is possible. The Apriori Algorithm is considered in detail; AIS, Eclat, FP-growth and the Fast Completion Apriori Algorithm are also studied. For each algorithm, we derive the probability that an item-set is a candidate (an item-set whose frequency status cannot be deduced by the algorithm) and that it is actually frequent or fails to be frequent.

1 Introduction

The frequent item-set problem, introduced in [2, 3], is a well known and interesting basic problem at the core of many data mining problems [1, 2, 3, 4, 9, 12, 14, 16]. The problem is, given a large database of basket data and a user-defined support threshold k, to determine which sets of items occur in at least k baskets. In the last two decades, several different algorithms for solving it were proposed [2, 3, 15, 18, 20]. In this paper, we focus on the frequent item-set algorithms that are candidate-based. For each algorithm, item-set I becomes a *candidate* if certain associated *test-sets* are already determined to be frequent. Based on these candidates, the frequent sets are found by counting the frequency of the item-set. We consider the Apriori Algorithm [3] in detail; the Agrawal-Imielinski-Swami (AIS) Algorithm [2], the Equivalence Class Transformation Algorithm (Eclat) [20], the Frequent Pattern Tree-growth (FP-growth) Algorithm [15] (which can be considered as a candidate-based algorithm, the title of [15] not withstanding; see also [12]) and the Fast Completion Apriori Algorithm are also studied but their analysis is similar to that of Apriori, so only the main principles are sketched. For the Apriori, AIS, Eclat, and FP-growth algorithms, the test-sets are item-sets that can be obtained by omitting a single item from I. For the Fast Completion Apriori Algorithm, the test-sets are all those subsets of I whose size is equal to the level where the regular Apriori Algorithm was last used.

For each of the above mentioned algorithms, we determine the probability that an item-set is a candidate and the probability that it is actually frequent, for a much more general model of shopping behaviour than in [19]. Our new shopping model can cover almost all situations that occur in reality; the shoppers are independently following a random shopping policy and each combination of items has its own probability, so any correlation in item purchasing can be accommodated. The probability that an item-set is a candidate depends on the particular algorithm that is used. On the contrary, for a given probability model of the data, all correct algorithms give the same probability that an item-set is frequent. This probability is called the *success probability*. The probability that an item-set is a candidate but not frequent, is called the *failure probability*. This probability is particularly

*The work of this author is supported by NSF grant IIS-0082407.

important because it is related to work that a better algorithm might hope to avoid. We compute the success and failure probability for all the algorithms and relate these probabilities to the probabilities of the corresponding test-sets for finding candidates. It turns out to be that the algorithms differ in which test-set is the most important one. We call the best test-set the one with the smallest probability. Our results show that for the Apriori and the Fast Completion Apriori Algorithm, the most important test-set is the best test-set. For the AIS Algorithm, it is the worst test-set. For Eclat-like algorithms (including FP-growth) it is a set that is at least as good as the second-worst test-set.

A survey of the best known frequent item-set algorithms can be found in [8] and [12]. The notion of candidates was introduced in [13] in a slightly different way. We already referred to other directly related work [19], where Apriori is considered in detail for the uniform shopping model.

Our main contributions are: (1) a probabilistic study of various candidate-based frequent item-set algorithms using a general random shopping model with arbitrary buying patterns, (2) the introduction and significance of the conceptual notion of test-sets for the determination of candidates, (3) the determination of the success and failure probability of an item-set I, (4) the discovery that the failure probability of item-set I is almost always determined by the probability of a single test-set of I and (5) the conclusion that the found probabilistic results are *sharp*: the algorithms may perform close to each other for certain data sets and for other data sets not.

In the next section, we describe the different algorithms considered in the paper. In the following section, the probability model is discussed in detail. After that, we give a proof of the claims about the Apriori Algorithm. Finally, we outline the proof of the claims for the other algorithms. We conclude with a discussion.

2 Preliminaries

2.1 The Apriori Algorithm

The Apriori Algorithm [3] uses the most powerful candidacy test: an item-set I is a candidate if and only if *each* of I's subsets obtained by removing one element is frequent. The algorithm processes the sets level-wise: it determines which sets of size 1 are frequent, then generates candidates of size 2 and determines which of these sets are frequent, etc. When processing level n, it already knows all frequent sets of size $n-1$. An item-set I of n items is a candidate for level n if *all* of its $n-1$ size subsets are frequent.

2.2 The Fast Completion Apriori Algorithm

We use the word "counting" to describe the examination of the data to determine whether a candidate is frequent, regardless of the details of how it is done. Usually, it is faster to count a batch of candidates rather than one candidate at a time. The Fast Completion Apriori Algorithm [3] uses this property. It starts out like the regular Apriori Algorithm, proceeding in a level-wise manner, generating and testing candidates as it goes, but as soon as it determines that the number of remaining candidates for all remaining levels is not too large, it generates candidates for the remaining levels based on the currently available information. So in fact, it runs the Apriori Algorithm but stops at level n. From that level on, the algorithms uses the frequent item-sets of size n to generate candidates for all remaining higher levels. An item-set I of size $n+h$ is a candidate for level $n+h$, if *all* of its subsets of size n are frequent.

2.3 The AIS Algorithm

The AIS algorithm [2] was the first algorithm introduced to solve the frequent item-set problem. The Apriori Algorithm is developed from the AIS Algorithm by strengthening the candidacy test. When the AIS Algorithm is doing level n processing, an item-set I is only considered when the algorithm comes across a basket that contains all the items of I. Thus, the set I is treated as a candidate if *any* size $n-1$ subset of I is frequent.

2.4 Eclat and FP-growth

As far as the test leading to candidacy, Eclat [20] and FP-growth [15] are the same algorithm, even though they have important differences in the details of how they organize the data for processing. They both build a tree of frequent item-sets based on an ordering of the items. Max-Miner [6] is almost the same algorithm. There are three different orderings that are commonly discussed: least-frequent-first [6], most-frequent-first [15], and arbitrary (or lexicographic) order [17]. The first two orders can be based on global frequency (static) or on frequencies for the current subtree (dynamic). There is an initialization phase to take care of the item-sets with only one or two elements. For the candidate generation, an algorithm of this type generates an item-set I as a candidate only when *two* particular subsets of I are frequent. The two test-sets associated with I are the one obtained by omitting the last item from I and the one obtained by omitting the next to last item from I. Based on the tree associated with building the frequent item-sets, we call the first test-set the father test-set and the second one the special-uncle test-set. Both the father and the special-uncle test-set must be frequent before I can be a candidate.

3 Random Shopping Model

Our shopping model considers a set of shoppers that have identical, independent and random shopping policies. The more general situation where each shopper has his own policy can be also handled with this model so long as the shoppers arrive in random order and they shop independently; replace the original shoppers with a single shopper whose policy is an average of the policies of the original shoppers.

4 General Analysis

We focus on what a candidate-based frequent item-set algorithm does while processing a *single* item-set, I. To determine the *total* work done by the algorithm, one must sum the results of this section over *all* item-sets, and this is often hard to do. One can, however, get a lot of insight into the behavior of the algorithm by focusing on what is important with respect to a single item-set.

In general, a candidate-based algorithm, when it

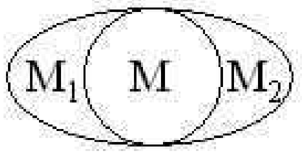

Figure 1: Graphical illustration of the ears associated with item-sets I_1 and I_2

is considering making item-set I a candidate, will have already counted some of the subsets of I. Figure 1 shows the situation in basket space when I is associated with two test-sets. Associated with I we have the region M with the baskets that contain all the items of I (along with perhaps additional items). If the two subsets of I are I_1 and I_2, then the left ear (labelled M_1) has those baskets with all the items of I_1 except those baskets that have all the items of I. Similarly, M_2 has all the baskets with the items of I_2 except those that have all the items of I. For all the algorithms we consider, except for the Fast Completion Apriori Algorithm, the various ears are disjoint. Item-set I will be a candidate if the number of baskets that are in either M or M_i, is at least k, for *every* test-set I_i.

We use the following notation in the analysis:

b, the total number of baskets in the database.

k, the support threshold. A frequent item-set algorithm determines which item-sets are contained in at least k baskets.

I, the item-set being considered. We name the elements of I with integers from 1 to $|I|$.

I_i, the i-th test-set associated with I. For the Apriori Algorithm, we have $I_i = I - \{i\}$ $(1 \leq i \leq |I|)$, but this is not true in general.

$P(I)$, the probability that a shopper buys all the items in I (regardless of whether or not other items are purchased). This definition implies $P(\emptyset) = 1$. Similarly, $P(I_i)$ is the probability that a shopper buys all the items in I_i. This is the probability that a basket is in the set of baskets associated with the i-th test-set I_i, $M \cup M_i$.

$Q_i(I) = P(I_i) - P(I)$, the probability that a shopper buys all the items in the i-th test-set without buying all the items in I. This is the probability that a basket contributes to the i-th test-set even though it does not contain all the items of I. In other words, the basket is in the ear associated with the i-th test-set, M_i, and $Q_i(I)$ is the probability of this ear M_i.

$Q(I)$, the $Q_i(I)$ that controls the performance of the algorithm.

l, the number of i values for which $Q_i(I) = Q(I)$. Thus, l is the multiplicity of the controlling value.

For item-set I, we compute:

$S(I)$, the success probability, i.e., the probability that at least k baskets contain all the items of I, so that I is frequent. Any correct algorithm for the frequent item-set problem has the same success probability. The success probability is a property of the data, not of the algorithm.

$F(I)$, the failure probability, i.e., the probability that item-set I passes the candidacy test but fails the frequency test. Some algorithms are faster than others because they have a smaller failure probability. The failure probability depends on both the problem instance and the algorithm.

The probability that item-set I is a candidate is $C(I) = S(I) + F(I)$.

The mathematics is similar to that in [19], but there are changes in the details so that we can handle the more general situation of this paper. Therefore, we concentrate on deriving the initial equations and explaining the notational transformations needed so that the proofs from the earlier paper will apply to the equations of this paper. We move quickly to the final results.

4.1 Success Probability

The probability that at least k of b shoppers have baskets that contain all of the items of set I is

$$S(I) = \sum_{j \geq k} \binom{b}{j} [P(I)]^j [1 - P(I)]^{b-j}. \qquad (1)$$

This probability of success applies to any correct frequent item-set algorithm.

When $P(I) \leq k/b$, define α_1 by

$$\alpha_1 = \frac{k}{b} - P(I). \qquad (2)$$

In this case $S(I)$ goes to 0 rapidly with increasing α_1. In particular, we can find

$$S(I) \leq e^{-b\alpha_1^2/\{2P(I)[1-P(I)]\} + O(b\alpha_1^3[1-P(I)]^{-2})} \qquad (3)$$

When $P(I) \geq (k-1)/b$, define α_2 by

$$\alpha_2 = P(I) - \frac{k-1}{b}. \qquad (4)$$

In this case $S(I)$ goes to 1 rapidly with increasing α_2. In particular, we can find

$$S(I) \geq 1 - e^{-b\alpha_2^2/\{2P(I)[1-P(I)]\} + O(b\alpha_2^3 P(I)^{-2})} \qquad (5)$$

4.2 Failure Probability

To compute the failure probability for an algorithm, we define the following conditions with respect to a single basket:

M: a shopper's basket contains all the items in I, and

M_i: a shopper's basket contains all the items of test-set I_i without containing all the items of I.

It is not important which items (if any) the shopper's basket contains among those not in item-set I. For all the algorithms we consider (except the Fast Completion Apriori Algorithm), these conditions are disjoint.

The probability that a shopper satisfies condition M is $P(I)$. The probability that a shopper satisfies condition M_i is $Q_i(I)$.

So long as the M_is are disjoint, the probability that j_0 shoppers satisfy condition M, j_1 shoppers satisfy condition M_1, …, j_n shoppers satisfy condition M_n and the remaining $b - j_0 - \cdots - j_n$ shoppers do not satisfy any of the conditions can be expressed by

$$\binom{b}{j_0, j_1, \ldots, j_n, b - j_0 - j_1 - \cdots - j_n}$$

$$\times [P(I)]^{j_0} \left[\prod_{1 \le i \le n} Q_i(I)^{j_i} \right]$$

$$\times \left[1 - P(I) - \sum_{1 \le i \le n} Q_i(I) \right]^{b - j_0 - \sum_{1 \le i \le n} j_i} \qquad (6)$$

If the M_i overlap, then the details are more complex, but the situation is similar and a variant of eq. (6) can be derived.

Thus, for any algorithm we are considering, eq. (6) (or a variant) gives the probability related to a particular set of counts (the j values). If this equation is summed over the cases that lead a particular algorithm to make I a candidate, then the formula gives the probability that I is a candidate. If we sum over the conditions that lead to I being a failure, then we obtain the probability that I is a failure. The set of conditions depend on the particular algorithm.

5 Failure Probability for the Apriori Algorithm

We now focus on the failure probability for the Apriori Algorithm. In this case, the number of tests associated with a set I is $|I|$. The test-sets are all the subsets of I, leaving one element out. Item-set I will be a candidate if the number of baskets that are in either M or M_i is at least k, for *every* test-set I_i. This happens when

$$j_0 + j_1 \ge k \text{ and } j_0 + j_2 \ge k \text{ and } \cdots$$

$$\text{and } j_0 + j_{|I|} \ge k \qquad (7)$$

is true. Thus, to find the probability that I is a candidate for the Apriori Algorithm, we must sum eq. (6) (with $n = |I|$) subject to condition (7), what leads to

$$C(I) =$$

$$\sum_{\substack{j_0 \\ j_1 \ge k - j_0 \\ j_2 \ge k - j_0 \\ \cdots \\ j_{|I|} \ge k - j_0}} \binom{b}{j_0, j_1, \ldots, j_{|I|}, b - j_0 - j_1 \cdots - j_{|I|}}$$

$$\times [P(I)]^{j_0} \left[\prod_{1 \le i \le |I|} Q_i(I)^{j_i} \right]$$

$$\times \left[1 - P(I) - \sum_{1 \le i \le |I|} Q_i(I) \right]^{b - j_0 - \sum_{1 \le i \le |I|} j_i} \qquad (8)$$

The subset of these cases, where j_0 is smaller than k, are the cases that lead to I being a failure. This is expressed in

$$F(I) = C(I) - S(I) =$$

$$\sum_{\substack{j_0 < k \\ j_1 \ge k - j_0 \\ j_2 \ge k - j_0 \\ \cdots \\ j_{|I|} \ge k - j_0}} \binom{b}{j_0, j_1, \ldots, j_{|I|}, b - j_0 - j_1 - \cdots - j_{|I|}}$$

$$\times [P(I)]^{j_0} \left[\prod_{1 \le i \le |I|} Q_i(I)^{j_i} \right]$$

$$\times \left[1 - P(I) - \sum_{1 \le i \le |I|} Q_i(I) \right]^{b - j_0 - \sum_{1 \le i \le |I|} j_i} \qquad (9)$$

5.1 Efficient Computation of $F(I)$

The number of operations needed to compute F by direct application of eq. (9) is $O(kb^{|I|})$. However, we can find a recurrence relation such that F can be computed in time that is polynomial in b and $|I|$, $O(|I|b^2)$.

5.2 Chernoff Bound for $F(I)$

For most values of $P(I)$ and the $Q_i(I)$, $F(I)$ will either be close to 0 or close to 1. Chernoff bounds [11] provide a good method to approximate $F(I)$.

For any $y \le 1$ and any set of $x_i \ge 1$, we have the Chernoff bound expressed by

$$F(I) \leq \Big[1+P(I)\Big(y \prod_{1\leq i\leq|I|} x_i-1\Big) + \sum_{1\leq i\leq|I|} Q_i(I)(x_i-1)\Big]^b y^{-k+1} \prod_{1\leq i\leq|I|} x_i^{-k} \quad (10)$$

To find the best Chernoff bound for F, the first step is to take the partial derivative of the logarithm of the right side of eq. (10) with respect to y and with respect to each x_i. Setting the partial derivative with respect to y to zero gives

$$(b-k+1)P(I)y \prod_{1\leq i\leq|I|} x_i - (k-1)\Big[1-P(I)+ \sum_{1\leq i\leq|I|} Q_i(I)(x_i-1)\Big] = 0. \quad (11)$$

Setting the partial derivative with respect to $x_{i'}$ to zero leads to

$$(b-k)P(I)y \prod_{1\leq i\leq|I|} x_i + bQ_{i'}(I)x_{i'} - k\Big[1-P(I)+ \sum_{1\leq i\leq|I|} Q_i(I)(x_i-1)\Big] = 0. \quad (12)$$

To find the optimum Chernoff bound, we rewrite eqs. (11) and (12). In the end, eq. (11) can be seen as

$$(b-k+1)P(I)yx^l - (k-1)[1-P(I)+lQ(I)(x-1)] = 0, \quad (13)$$

and eq. (12) as

$$(b-k)P(I)yx^l+bQ(I)x - k[1-P(I)+lQ(I)(x-1)] = 0. \quad (14)$$

When $Q(I) + P(I) \leq k/b$, define α_3 by

$$\alpha_3 = \frac{k}{b} - [P(I) + Q(I)]. \quad (15)$$

In this case, $F(I)$ goes rapidly to 0. In particular, when α_3 is small enough, i.e.,

$$\alpha_3 = \{lP(I) + Q(I) - l[Q(I) + P(I)]^2\}o(1) \quad (16)$$

and b is large enough that the second smallest $Q_i(I)$ is not important,

$$F(I) \leq e^{-bl\theta\alpha_3^2/(2\{Q(I)+P(I)+(l-1)P(I)-l[Q(I)+P(I)]^2\})} \quad (17)$$

gives an upper bound on the failure probability. Here θ is used to represent a function that approaches 1 in the limit.

When $P(I) \geq (k-1)/b$, $F(I)$ goes rapidly to zero because $F(I) \leq 1 - S(I)$. In particular, we can find

$$F(I) \leq e^{-b\alpha_2^2/\{2P(I)[1-P(I)]\}+O(\alpha_2^3 bP(I)^{-2})} \quad (18)$$

when α_2 is defined by eq. (4).

To obtain a lower bound on $F(I)$, we use inclusion-exclusion arguments leading to

$$F(I) \geq 1-e^{-b\alpha_1^2/\{2P(I)[1-P(I)]\}+O(b\alpha_1^3[1-P(I)]^{-2})} - \sum_{1\leq i\leq|I|} e^{-b\beta_i^2/\{2[P(I_i)][1-P(I_i)]\}+O(b\beta_i^3[P(I_i)]^{-2})} \quad (19)$$

where $P(I_i) = P(I) + Q_i(I)$ and where α_1 and β_i related to k by $k = b[P(I) + \alpha_1]$ and $k = b[P(I) + Q_i(I) - \beta_i] + 1$ when α_1 and all the βs are positive. For those cases where α_1 or some β is too large for the big O term to be small, one can still obtain a correct bound by replacing the too-large parameter with a smaller value. So long as α_1 and none of the βs are near zero, the value of F is near 1.

The diagram in Figure 2 is useful for understanding the behavior of the Apriori Algorithm with regard to a single item-set, I, where the associated test-sets are $I_1, \ldots, I_{|I|}$. Item-set I has probability $P(I)$. Each test-set I_i has some larger (or equal) probability because each basket that has all the items of I also has all the items of each test-set I_i. If the fractional threshold k/b is less than $P(I)$, the probability of buying all the items of I, then item-set I is nearly always a success, so frequent. In this case, the failure probability is almost zero. If k/b is between $P(I)$ and $P(I_j)$ (where I_j is the best test-set, i.e., the one with smallest probability, then the probability that I is a success is almost zero, but the probability that it is a candidate is almost one, so the probability that item-set I is a failure for the Apriori Algorithm is almost one. If k/b is above $P(I_j)$, then there is almost no chance that I is even a candidate, so both the probability of failure and

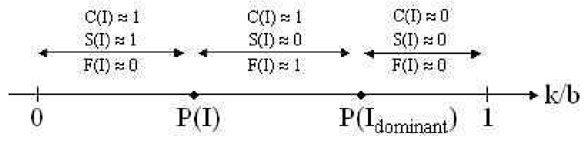

Figure 2: Whether item-set I is frequent (a success), or a candidate which is infrequent (a failure), or not even a candidate, is determined (with high probability) by where the threshold ratio k/b falls in relation to the probability of item-set I and the probability of I's most important test-set $I_{dominant}$.

the probability of success are nearly zero. Nearly all the failures for the Apriori Algorithm, i.e. counting item-sets which are not frequent, come from those sets where the set's probability is below the threshold but where the probability of the best test-set is above the threshold.

6 Failure Probability for Other Algorithms

In this section, we consider the failure probability for the remaining algorithm. For AIS, Eclat, and FP-growth, the relevant test-sets for item-set I are still the item-sets I_i, with one item less. For the Fast Completion Apriori Algorithm, the relevant test-sets are all the subsets of I that have size n, where n is the last level where the regular Apriori Algorithm was used.

6.1 AIS

For AIS, a set I is a candidate if (1) I occurs in a basket and (2) *some* subset of I that is missing one item is frequent. For a simple analysis, we take the viewpoint that an item-set I is a candidate if it satisfies the second condition and that the time spent in counting is proportional to the number of baskets that I is in. Thus, if I meets the second condition but not the first, then it is a candidate that requires no time. We also briefly indicate how to do the more complex analysis where both conditions must be true for I to be a candidate.

With the simple approach, item-set I will be a candidate exactly when the condition

$$j_0+j_1 \geq k \text{ or } j_0+j_2 \geq k \text{ or } \cdots \text{ or } j_0 + j_{|I|} \geq k \tag{20}$$

is true. It will be a failure when eq. (20) is true with $j_0 < k$. Thus, the failure probability for AIS is given by eq. (9) except that the condition on the summation is

$$j_0 < k \text{ and } (j_0+j_1 \geq k \text{ or } j_0+j_2 \geq k \text{ or } \cdots \text{ or } j_0 + j_{|I|} \geq k). \tag{21}$$

Inclusion-exclusion type reasoning can be used on the logical or operation. The sum is equivalent to the sum of a bunch of subsums (some occurring with a negative sign). The i-th subsum of the first group has the condition

$$j_0 < k \text{ and } j_0 + j_i \geq k \tag{22}$$

(in the subsum, each index not mentioned in condition (22) must be summed-out by summing over all possible values of the index). There is a second group of subsums that must be subtracted with the condition

$$j_0 < k \text{ and } j_0 + j_{i_1} \geq k \text{ and } j_0 + j_{i_2} \geq k \tag{23}$$

because we have double counted the situations where two j_i's satisfy eq. (20). These groups continue until we run out of corrections. From the detailed analysis that we did for the Apriori Algorithm, we can approximate all of these subsums. The only subsum that is important is the one associated with the weakest of the test-sets, the test-set with the highest probability, so long as the largest $Q_i(I)$ is not too close to the others. Thus, when we have a unique worst test-set (no near ties), the failure probability for the AIS Algorithm is bounded by eqs. (17), (18), and (19) with $Q(I)$ being the largest of the $Q_i(I)$ and $l = 1$. The performance of AIS is in this case determined by the worst test-set. If I_m is the size $m - 1$ subset (of a candidate set I of size m) with *largest* probability of being purchased, then I is a candidate with probability near 1 when the probability of buying all the items of I_m is significantly above k/b, and it is near 0 when the probability of buying the items is significantly below

k/b. The other test-sets have only a slight effect on the probability that item-set I will be a candidate. When several item-sets tie for worst, the bound on the failure probability is worse than that given by eqs. (17), (18), and (19) but only by a factor that is no larger than the number of ties. Whether $F(I)$ is near 0 or 1 still depends on whether or not k/b is between $P(I)$ and $P(I_i)$, but now I_i is the worst test-set associated with I.

If we want to analyze the effect of requiring a candidate to have an associated basket, then we need to reduce the failure probability from the previous calculation by allowing for the cases where item-set I does not occur in any basket. When k is larger than 1, the correction is usually not important.

6.2 Eclat and FP-growth

Eclat and FP-growth can be treated as the same algorithm, even though they have important differences in how they organize the data for processing [see Section 2.4]. Since both the father and the special-uncle must be frequent, the best of those two has the main effect on whether I is a candidate. Thus, the failure probability for the algorithm is bounded by eqs. (17), (18), and (19) with $Q(I)$ equal to the smaller of the $Q_i(I)$ for the father and the $Q_i(I)$ for the special-uncle. The value of l is 1 when these two Qs are different, and l is 2 when they are equal.

Which two Qs control the failure probability depends on the ordering that is used. If the order is dynamic with the most-frequent item first, then the father test-set will be the worst test-set and the special uncle test-set will be the second worst. If the order is static with the most-frequent item first, then there will be a strong tendency for the father test-set to be the worst one and for the special-uncle test-set to be the second-worst one. If this happens, the second-worst test-set has the main effect on the probability of candidacy, but the actual situation will depend on what correlations are present in the data. If the order is dynamic with the least-frequent item first, then the father test-set will be the best test-set and has the main effect. If the order is static with the least-frequent item first, the father test-set will likely be the best test-set. If the items are choosen in a random lexicographic order, both the parent and the special uncle will each be randomly-choosen test-sets. Each pair of possible Qs is thus equally likely to be chosen and the best of the two randomly-chosen test-sets will have the main effect. Once we have determined which test-set has the main effect, the situation is the same as for the previous algorithms; the item-set I is a candidate with probability near 1 when the probability of buying all the items of the important test-set is significantly above k/b and it is near 0 when the probability of buying is significantly below k/b.

On many data sets dynamic least-frequent-first Eclat has essentially the same performance (generated and count the same candidates) as Apriori, but there are interesting cases with anti-correlated data where this is not the case. Suppose items have the following frequency ordering: $a < b < c$ (let a = hot dog, b = Pepsie Cola and c = Coca Cola) and the following doubleton frequency ordering $bc < ab < ac$. For thresholds in the range $bc < k < ab$, Eclat counts the set abc but Apriori does not.

6.3 Fast Completion Apriori Algorithm

The test-sets for an item-set I are its subsets consisting of n elements, where n is the last level where the regular Apriori Algorithm is used. Unlike the previous cases, we now have overlapping test-set ears. Again, the performance of the algorithm on item-set I is determined primarily by the best of I's test-sets. Of course, this best test-set will come from the level where the regular Apriori Algorithm stopped, so it usually will be a much worse test-set than the one the Apriori Algorithm would use.

Rather than do a detailed calculation of the effect of the overlap, we will use some simple ideas to show that eqs. (17), (18), and (19) still bound the failure probability with $Q(I)$ being the $Q_i(I)$ for the best test-set and $l = 1$, so long as there is no tie for the best Q.

The presence of overlapping ears reduces the effectiveness of the test-sets compared to the no-overlap case. However even in the presence of overlapping ears the collection of test-sets is at least as effective as the best of the test-sets in the collection of test-sets. When l (the number of ties for best) is one, the bounds are all in terms of the Q for the best test-set. Thus, they still apply even when there are overlaps. When there is a tie for best, a more complex analysis is needed to obtain the best possible bounds, but the previous bounds still apply if one uses a modified l, where the modified l is somewhere between 1 and the original l.

The only significant weakness of the Fast Completion Apriori Algorithm with regard to generating candidates is that the test-sets used come from several levels back and thus are likely to have much higher probabilities than test-sets from one level back would have.

7 Conclusion

In this paper, we discuss algorithms that have a candidacy test that considers an item-set I when only some of I's subsets are frequent. The various algorithms mentioned differ in which subsets they consider for the candidacy test. It is clear that an algorithm that considers *all* subsets that are missing one item (such as Apriori) will sometimes need to consider fewer candidates than those that have a weaker candidacy test, but it is less obvious how significant these differences are. We use probabilistic analysis to show that the significance depends both on the algorithm and on properties of the data set that the algorithm is being run on (and in the case of Eclat-like algorithms, the ordering assumption).

For *random* data, all the algorithms considered have about the same performance, because all test-sets of a given size have about the same frequency. So, least-frequent-first Eclat-like algorithms perform on most data sets nearly as well as the Apriori Algorithm, since, in both cases, the candidacy of an item-set is essentially determined by frequency status of its best test-set.

For *highly correlated data*, the algorithms differ greatly in their performance. In this case, the various subsets of I with size one less may have hughly different frequencies. The performance of the algorithms depends on whether or not their candidancy test includes the most important subset, the one whose frequency is expected to be closest to the frequency of I. Those algorithms with the candidacy test which includes the most important test-sets have essentially the same performance. Those that do not have much worse performance. When the Eclat-like algorithms use a least-frequent-first order, they perform nearly as well as the Apriori Algorithm unless the data set has strongly anti-correlated data. When the Eclat-like algorithms use a random order, their performance can be significantly worse than that of the Apriori algorithm, but significantly better than the performance of the AIS algorithm. When they use a most-frequent-first ordering, the Eclat-like algorithms perform nearly as poorly as the AIS-algorithm, since in this cases, the candidacy of an itemset is essentially determined by the frequency status of its worst test-set.

Acknowledgments

We thank Toon Calders and Bart Goethals for helpful discussions on frequent item-set algorithms.

References

[1] Kamal Ali, Stefanos Manganaris, Ramakrishnan Srikant. Partial Classification using association rules. In *Proc. KDD Int. Conf. Knowledge Discovery in Databases*, pages 115–118, 1997.

[2] Rakesh Agrawal, Tomasz Imielinski, and Arun Swami. Mining association rules between sets of items in large databases. In *Proc. ACM SIGMOD Int. Conf. Management of Data*, pages 207–216, Washington, D.C., 1993.

[3] Rakesh Agrawal and Ramakrishnan Srikant. Fast Algorithms for Mining Association Rules. In *Proc. of the 1994 Very Large Data Bases Conference*, pages 487–499, 1994.

[4] Rakesh Agrawal and Ramakrishnan Srikant. Mining Sequential Patterns. In *Proc. IEEE ICDE Int. Conf. on Data Engineering*, pages 3–14, 1995.

[5] Yves Bastide, Rafik Taouil, Nicolas Pasquier, Gerd Stumme, Lofti Lakhal. Mining Frequent Patterns with Counting Inference. In *SIGKDD Explorations,* **2***(2)*, pages 66-75, 2000.

[6] Roberto J. Bayardo. Efficiently Mining Long Patterns from Databases. In *Proc. of the 1998 ACM SIDMOD Int. Conf. on Management of Data*, pages 85–93, 1998.

[7] Douglas Burdick, Manuel Calimlim, and Johannes Gehrke. MAFIA: A Maximal Frequent Itemset Algorithm for Transactional Databases. In *Proc. of the 17th Int. Conf. on Data Engineering*, pages 443–452, 2002.

[8] Toon Calders. Computational Complexity of Itemset Frequency Satisfiability. In *Proc. of the 23rd ACM SIGACT-SIGMOD-SIGART Symp. on Principles of Database Systems*, pages 143–154, 2004.

[9] Toon Calders and Bart Goethals. Mining all Non-Derivable Frequent Itemsets. In *Proc. of the 6th European Conf. on Principles and Practice of Knowledge Discovery in Databases (PKDD'02), Lecture Notes in Artificial Intelligence.*, pages 74–85, 2002.

[10] Herman Chernoff. A Measure of Asymptotic Efficiency for Test of a Hypothesis Based on the Sum of Observations. In *Annals of Mathematical Statistics,* **23**, pages 493–507, 1942.

[11] Floris Geerts and Bart Goethals and Jan Van den Bussche. A Tight Upper Bound on the Number of Candidate Patterns. In *Proc. of the first IEEE Int. Conf. on Data Mining*, San Jose, CA, USA2001.

[12] Bart Goethals. Survey of Frequent Pattern Mining. *[http://www.adrem.ua.ac.be/∼goethals/publications.html]*, 2003

[13] Bart Goethals. Efficient Frequent Pattern Mining. *PhD thesis*, transnational University of Limburg, Diepenbeek, Belgium, December 16 2002.

[14] Bart Goethals and Mohammed J. Zaki (eds.). Proc. of the Workshop on Frequent Itemset Mining Implementations, Melbourne, Florida, 2003.

[15] Jiawei Han, Jian Pei, Yiwen Yin. Mining Frequent Patterns without Candidate Generation In *Proc. of the 2000 ACM SIGMOD Int. Conf. Management of Data*, Dallas, TX, pages 1–12, 2000.

[16] Jochen Hipp, Ulrich Güntzer, Gholamzera Nakhaeizadeh. Algorithms for Association Rule Mining — A General Survey and Discovery of Association Rules — A Position Paper. In *ACM SIGKDD Explorations,* **2**, pages 65–69.

[17] Walter A. Kosters, Wim Pijls. Apriori, A Depth First Implementation. In *Proc. of the Workshop on Frequent Itemset Mining Implementations*, Melbourne, Florida, 2003.

[18] Heikki Mannila and Hannu Toivonen. Levelwise Search and Borders of Theories in Knowledge Discovery. In *Data Mining and Knowledge Discovery,* **1**, pages 241–258, 1997.

[19] Paul W. Purdom, Dirk Van Gucht, and Dennis P. Groth. Average Case Performance of the Apriori Algorithm. In *SIAM J. Computing,* **33***(5)*, pages 1223–1260, 2004.

[20] Mohammed J. Zaki. Scalable Algorithms for Association Mining. In *IEEE Transactions on Knowledge and Data Engineering,* **12***(3)*, pages 372–390, 2000.