A Probability Analysis for Candidate-Based Frequent Itemset Algorithms

Nele Dexters
University of Antwerp
Middelheimlaan 1
2020 Antwerp, Belgium
nele.dexters@ua.ac.be

Paul W. Purdom
Indiana University
Computer Science
Bloomington, Indiana, 47405
pwp@cs.indiana.edu

Dirk Van Gucht*
Indiana University
Computer Science
Bloomington, Indiana, 47405
vgucht@cs.indiana.edu

ABSTRACT

This paper explores the generation of candidates, which is an important step in frequent itemset mining algorithms, from a theoretical point of view. Important notions in our probabilistic analysis are success (a candidate that is frequent), and failure (a candidate that is infrequent). For a selection of candidate-based frequent itemset mining algorithms, the probabilities of these events are studied for the shopping model where all the shoppers are independent and each combination of items has its own probability, so any correlation between items is possible. The Apriori Algorithm is considered in detail; for AIS, Eclat, FP-growth and the Fast Completion Apriori Algorithm, the main principles are sketched. The results of the analysis are used to compare the behaviour of the algorithms for a variety of data distributions.

1. INTRODUCTION

The frequent itemset mining (FIM) problem [1, 2] is a well-known basic problem at the core of many data mining problems [3, 5, 6]. The problem is, given a database of basket data and a user-defined support threshold k, to determine which sets of items are bought by at least k shoppers, so occur in at least k baskets.

Since its introduction, several different algorithms for solving the problem were proposed [4, 5, 13], and experimentally analyzed [5, 14]. A survey of the best known FIM algorithms can be found in [9]. In comparison to the extensive literature on the experimental analysis, relatively few papers have been devoted to theoretical analyses [8, 12]. Our paper adds to the theoretical aspect of the FIM problem.

The aim of this research is an analytical study of the properties of the FIM problem and the performance of the associated algorithms and their interaction with different data distributions. The number of candidates has a major effect on the computation work done by the algorithm. Therefore, in the context of a general probabilistic model where all the shoppers shop independently of each other and each combination of items has its own probability of being purchased (so any correlation between items is possible), we theoretically analyze the probability that an itemset is a *candidate* for a variety of candidate-based FIM algorithms: the Apriori Algorithm [2], which we will analyze in detail, AIS [1], Eclat [13], FP-growth [10] (which can be considered as a candidate-based algorithm, the title of [10] not withstanding) and the Fast Completion Apriori (FCA) Algorithm [2]. Their analysis is similar to that of Apriori, so only the main principles are sketched. For these algorithms, an itemset becomes a candidate if certain associated *testsets* are already determined to be frequent. For the Apriori, AIS, Eclat, and FP-growth algorithms, the testsets are itemsets that are obtained by omitting a single item from I; for FCA, the testsets are all those subsets of I whose size is equal to the level where the regular Apriori Algorithm was last used. For example, itemset *abcd* is a candidate for Apriori when all its subsets *abc*, *abd*, *acd* and *bcd* are frequent; for Eclat, the same set *abcd* might be a candidate when *abc* and *abd* are frequent. Because of this candidacy definition that differs for each algorithm, the candidacy probability is dependent of the particular algorithm used. Once all candidates are found, their exact frequency status has to be counted explicitly in the database. If a considered candidate set turns out to be frequent, it is called a *success*; otherwise, it is a *failure*. All correct FIM algorithms have the same success probability; the failure probability is algorithm dependent and is particularly important because it is related to work that a better algorithm might hope to avoid. We compute the candidacy, success and failure probabilities for all the algorithms. It turns out that the algorithms differ in which testset is the most important one.

In [12], Apriori is considered in detail for the uniform shopping model. Our paper differs in two significant ways. Our new shopping model is much more general and covers almost all realistic situations. Furthermore, the study performed in our paper is not only for Apriori, but for several other algorithms and is based on the conceptual notion of candidate sets and testsets and their probabilistic relationship for being frequent.

Our theoretical approach is useful for practical purposes, for example for designers and implementors of FIM algorithms or for algorithms based on FIM, such as Association Rule Mining Algorithms.

The main contributions of this research are: (1) the deriva-

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

SAC'06 April 23-27, 2006, Dijon, France

tion of the candidacy, success and failure probabilities of itemsets for various candidate-based FIM algorithms, using a general random shopping model with arbitrary buying patterns that can cover associations, (2) the failure probability of an itemset I is essentially determined by the probability of a particular testset of I, obtained by omitting one element of I, but which testset this is depends on the algorithm, and (3) the comparison of the behavior of these candidate-based FIM algorithms for a variety of data distributions.

2. CANDIDATE-BASED FIM ALGORITHMS

2.1 The AIS Algorithm

The AIS algorithm [1] is the first algorithm introduced to solve the FIM problem. An itemset I of size n is treated as a candidate if *any* $n-1$ size subset of I is frequent.

2.2 The Apriori Algorithm

The Apriori Algorithm [2] works 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 $n-1$. Apriori is developed from AIS by strengthening the candidacy test: an itemset I consisting of n items is a candidate if *all* of its $n-1$ size subsets are frequent.

2.3 The Fast Completion Apriori Algorithm

The FCA Algorithm [2] 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 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 a certain level n. From that level on, the algorithm uses the frequent itemsets of size n to generate candidates for all remaining higher levels. An itemset I of size $n+h$ is a candidate if *all* of its subsets of size n are frequent.

2.4 Eclat and FP-growth

In [9], it is shown that Eclat [13] and FP-growth [10] are candidate-based algorithms. As far as the test leading to candidacy, they are essentially the same algorithm [9], even though they have important differences in the details of how they organize the data for processing.

Both Eclat and FP-growth build a tree of frequent itemsets based on an *ordering* of the items (cfr. Max-Miner [4]). There are three different orderings that are commonly used: least-frequent-first (LFF) [4], most-frequent-first (MFF) [10], and arbitrary or lexicographic order [11]. Both LFF and MFF can be applied as a *static* ordering, based on global frequencies, or as a *dynamic* ordering, based on frequencies for the current subtree.

An itemset I is a candidate only when *two* particular subsets of I are frequent. These two testsets are the one obtained by omitting the last item from I (the father testset) and the one obtained by omitting the next-to-last item from I (the special-uncle testset).

3. GENERAL ANALYSIS

3.1 Shopping Model

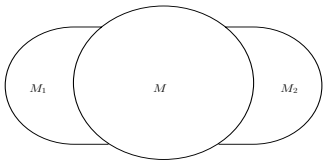

Figure 1: Graphical illustration of the region M associated with itemset I, and the ears M_1 and M_2 associated with testsets I_1 and I_2, respectively.

The shopping model considers a set of shoppers that have identical, independent and random shopping policies: all the shoppers behave the same and shop independently of each other; each combination of items in the shop has its own probability of being purchased. This model is very general: we can accomodate the existence of dependencies. In this approach, we assume that the shoppers do not change their policy over time.

The more general situation where each shopper has his own policy can also be handled with this model, as long as the shoppers arrive in random order and as long as they make their choices independently. The idea is to replace the original shoppers by a *single* one whose policy is a weighted linear combination of the policies of the original shoppers.

3.2 Analysis

This paper focusses on what a candidate-based FIM algorithm does while processing a *single* itemset, I. To determine the *total* work done by the algorithm, the results of this section have to be summed over *all* itemsets. For the general random shopping model used in this paper, this is hard to do. One can, however, gain a lot of insight into the behavior of the algorithms by focusing on what is important with respect to a single itemset.

In general, a candidate-based algorithm, when it is considering making itemset I a candidate, will have already counted some of the subsets of I. These sets are going to be used as *testsets*. Figure 1 shows the situation in *basket space* when I is associated with two testsets, I_1 and I_2. The region M contains the baskets that contain all the items of I (along with perhaps additional items). The left ear M_1 contains the baskets that contain all the items of I_1, except for those baskets that are in M (so contain all the items of I). Similarly, M_2 contains all the baskets with the items of I_2 except those that have all the items of I (and are in M). For all the algorithms we consider (except for FCA), the various ears are disjoint. Itemset I is a candidate if the number of baskets that are in $M \cup M_i$ is at least k, for *every* testset I_i.

We use the following notations in the analysis:

b, the total number of baskets in the database.

k, the support threshold. A FIM algorithm determines which itemsets are contained in at least k baskets.

I, the itemset being considered. The elements of I are named with integers from 1 to $|I|$.

I_i, the i-th testset 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; $P(I)$ is the probability that a basket is in M. This definition implies $P(\emptyset) = 1$. Similarly, $P(I_i)$ is the probability that a shopper buys all the items in I_i; $P(I_i)$ is the probability that a basket is in the set of baskets associated with the i-th testset 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 testset without buying all the items in I. This is the probability that a basket contributes to the i-th testset 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 testset, M_i, and $Q_i(I)$ is the probability of this ear.

$Q(I)$, the $Q_i(I)$ that dominates in the determination of the magnitude of the failure probability of itemset I.

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

For itemset 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 FIM problem has the same success probability. It is a property of the data, not of the algorithm.

$F(I)$, the *failure* probability, i.e., the probability that itemset 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 itemset I is a candidate is

$$C(I) = S(I) + F(I).$$

The mathematics are similar to that in [12], but there are changes in the details so that we can handle the more general situation of this paper.

3.3 Success Probability

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

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

When $P(I) \leq k/b$, it can be shown that

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

with $\alpha_1 = k/b - P(I)$. In this case $S(I)$ goes to 0 rapidly with increasing α_1.

When $P(I) \geq (k-1)/b$, we can find

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

with $\alpha_2 = P(I) - (k-1)/b$. In this case $S(I)$ goes to 1 rapidly with increasing α_2.

3.4 Candidate and Failure Probability

To compute the candidate and failure probability, 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 testset I_i without containing all the items of I.

For all the algorithms we consider (except FCA), these conditions are disjoint.

The probability that a shopper satisfies condition M is $P(I)$, the probability that he 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

$$\binom{b}{j_1, \ldots, j_n, b - j_0 - j_1 - \cdots - j_n} \times [P(I)]^{j_0} \left[\prod_{1 \leq i \leq n} Q_i(I)^{j_i} \right] \times \left[1 - P(I) - \sum_{1 \leq i \leq n} Q_i(I) \right]^{b - j_0 - \sum_{1 \leq i \leq n} j_i}. \quad (4)$$

If the M_i overlap, the details are more complex, but the situation is similar and a variant of eq. (4) can be derived.

Thus, for any algorithm we are considering, eq. (4) (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 depends on the particular algorithm.

4. FAILURE PROBABILITY FOR APRIORI

Itemset I is a candidate if for every testset I_i ($1 \leq i \leq |I|$), the number of baskets that are in $M \cup M_i$ is at least k. This happens when

$$j_0 + j_1 \geq k \text{ and } j_0 + j_2 \geq k \text{ and } \cdots \text{ and } j_0 + j_{|I|} \geq k \quad (5)$$

is true. Thus, to find the probability that I is a candidate for the Apriori Algorithm, we must sum eq. (4) (with $n = |I|$) subject to condition (5). The subset of these cases, where j_0 is smaller than k, are the cases that lead to I being a failure:

$$F(I) = C(I) - S(I) = \sum_{\substack{j_0 < k \\ j_1 \geq k - j_0 \\ j_2 \geq k - j_0 \\ \cdots \\ j_{|I|} \geq k - j_0}} \binom{b}{j_0, j_1, \ldots, j_{|I|}, b - j_0 - j_1 - \cdots - j_{|I|}} \times [P(I)]^{j_0} \left[\prod_{1 \leq i \leq |I|} Q_i(I)^{j_i} \right] \times \left[1 - P(I) - \sum_{1 \leq i \leq |I|} Q_i(I) \right]^{b - j_0 - \sum_{1 \leq i \leq |I|} j_i}. \quad (6)$$

4.1 Chernoff Bound for F(I)

Chernoff bounds [7] provide a good method to approximate $F(I)$.

When $Q(I) + P(I) \leq k/b$, it can be shown that

$$F(I) \leq e^{-bl\theta\alpha_3^2/(2\{Q(I)+P(I)+(l-1)P(I)-l[Q(I)+P(I)]^2\})} \quad (7)$$

with $\alpha_3 = k/b - [P(I) + Q(I]$; θ is used to represent a function that approaches 1 in the limit. In this case, $F(I)$ goes rapidly to 0.

When $P(I) \geq (k-1)/b$, $F(I)$ goes rapidly to 0 because $F(I) \leq 1 - S(I)$. In particular, we can find

$$F(I) \leq e^{-b\alpha_2^2/\{2P(I)[1-P(I)]\}+O(\alpha_2^3 bP(I)^{-2})} \quad (8)$$

when $\alpha_2 = P(I) - (k-1)/b$.

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

$$F(I) \geq 1 - e^{-b\alpha_1^2/\{2P(I)[1-P(I)]\}+O(b\alpha_1^3[1-P(I)]^{-2})} - \sum_{1\leq i\leq |I|} e^{-b\beta_i^2/\{2[P(I_i)][1-P(I_i)]\}+O(b\beta_i^3[P(I_i)]^{-2})} \quad (9)$$

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.

4.2 Interpretation

The diagram in Figure 2 is illustrating the behavior of Apriori with regard to a single itemset, I, where the associated testsets are I_1, ..., $I_{|I|}$. Itemset I has probability $P(I)$. Each testset I_i has some larger (or equal) probability because each basket that contains all the items of I also contains all the items of each testset I_i. If the fractional threshold k/b is less than $P(I)$, then itemset I is always a success [$S(I) \approx 1$]. In this case, the failure probability is (almost) zero [$F(I) \approx 0$]. If k/b is above $P(I_{dominant})$ (where $I_{dominant}$ is the *best* testset, i.e., the one with the smallest probability), then there is almost no chance that I is even a candidate [$C(I) \approx 0$], so both the success and failure probabilities are (nearly) zero [$S(I) \approx 0$, $F(I) \approx 0$]. If k/b is between $P(I)$ and $P(I_{dominant})$, the probability that I is a candidate is almost one [$C(I) \approx 1$], but the probability that I is a success is almost zero [$S(I) \approx 0$], so the probability that I is a failure is almost one [$F(I) \approx 1$].

Nearly all the failures for the Apriori Algorithm, i.e. counting itemsets which are not frequent, come from those sets where the set's probability, $P(I)$, is below the threshold k/b but where the probability of the *best* testset, $P(I_{dominant})$, is above the threshold k/b.

5. FAILURE PROBABILITY FOR OTHER ALGORITHMS

5.1 AIS

For AIS, an itemset I is a candidate if I occurs in a basket and some subset of I that is missing one item is frequent. For a simple analysis, we only take care of the second condition. Itemset I is a candidate when the condition

$$j_0 + j_1 \geq k \text{ or } j_0 + j_2 \geq k \text{ or } \cdots \text{ or } j_0 + j_{|I|} \geq k \quad (10)$$

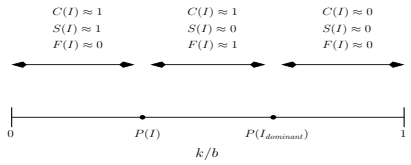

Figure 2: Whether itemset I is frequent (a success), or a candidate which is infrequent (a failure), or not even a candidate, is determined (with high probability) by where the threshold ratio k/b falls in relation to the probability of itemset I and the probability of I's most important testset $I_{dominant}$.

is true. It is a failure when eq. (10) is true with $j_0 < k$. Thus, the failure probability for AIS is given by eq. (6) except that the condition on the summation is

$$j_0 < k \text{ and } (j_0 + j_1 \geq k \text{ or } j_0 + j_2 \geq k \text{ or } \cdots \text{ or } j_0 + j_{|I|} \geq k). \quad (11)$$

Inclusion-exclusion type reasoning can be used on the logical or operation: the sum is equivalent to the sum of a collection of subsums, some occurring with a negative sign. From the detailed analysis that we did for the Apriori Algorithm, we can approximate all of these subsums individually.

Because an itemset I is a candidate for AIS if *any* $n-1$ size subset of I is frequent, the only subsum in this collection of inclusion-exclusion subsums that is important is the one associated with the *weakest* of the testsets, the testset 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 testset (no near ties), the failure probability for the AIS Algorithm is bounded by eqs. (7) , (8), and (9) with $Q(I)$ being the largest of the $Q_i(I)$ and $l = 1$.

The performance of AIS is determined by the *worst* testset. If I_m is the $m-1$ size 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 testsets have only a slight effect on the probability that itemset I will be a candidate. When several itemsets tie for worst, the bound on the failure probability is worse than that given by eqs. (7), (8), and (9) 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 $P(I_m)$. Figure 2 is again applicable, but with $I_{dominant} = I_m$.

5.2 Fast Completion Apriori Algorithm

The testsets for an itemset 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 testset ears. The presence of overlapping testset ears reduces the effectiveness compared to the no-overlap case but we can show that the performance of the algorithm on itemset I is determined primarily by the *best* of I's testsets. Of course, this best testset comes from the level where the regular Apriori Algorithm stopped (several levels back), so it usually is a much worse testset than the one Apriori would use.

Eqs. (7), (8), and (9) still bound the failure probability

with $Q(I)$ being the $Q_i(I)$ for the best testset and $l = 1$, so long as there is no tie for the best Q.

5.3 Eclat and FP-growth

Since both the father and the special-uncle must be frequent for a set I to be a candidate, the *best* of those two testsets has the main effect on whether I is a candidate. Thus, the failure probability is bounded by eqs. (7), (8) and (9) 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 MFF, the father testset will be the worst testset and the special uncle testset will be the second worst. If the order is static MFF, there will be a strong tendency for the father testset to be the worst one and for the special-uncle testset to be the second-worst one. If this happens, the *second-worst* testset 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 LFF, the father testset will be the *best* testset and has the main effect. If the order is static LFF, the father testset will likely be the best testset. If the items are chosen in a random lexicographic order, both the parent and the special uncle will each be randomly-chosen testsets. Each pair of possible Qs is thus equally likely to be chosen and the *best* of the two randomly-chosen testsets will have the main effect.

Once we have determined which testset has the main effect, the situation is the same as for the previous algorithms, as illustrated in Figure 2; the itemset I is a candidate with probability near 1 when the probability of buying all the items of the important testset is significantly above k/b and it is near 0 when the probability of buying is significantly below k/b.

6. DISCUSSION

In this paper, we discuss algorithms that have a candidacy test that considers an itemset I when only *some* of I's subsets, the testsets, are frequent. The various algorithms 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. We can show this with an example of anti-correlated data. Suppose items have the following frequency ordering: $a < b < c$ (let a = hot dog, b = Pepsi 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. It is clear that Apriori does less work compared to Eclat, but it is less obvious *how* significant these differences are. We use the probabilistic analyses from the previous sections to show that the significance depends both on the *algorithm* and on *properties of the data set* being processed (and in the case of Eclat-like algorithms, the *ordering assumption*).

The testsets for Eclat are a subset of the testsets for Apriori and a superset of the testsets for AIS. Therefore, the performance of Eclat is intermediate between those of Apriori and of AIS.

For data that are suitable *uniform* (the various testsets of the same size have similar probabilities), all the algorithms (other than FCA) have essentially the same performance in terms of the number of candidates generated.

If the data is non-uniform but suitably *independent* and *random*, then Apriori has essentially the same performance as LFF Eclat; Random Eclat is worse; MFF Eclat is worse yet and AIS is worst of all.

When the data has strong *anti-correlations*, Apriori is the best, followed by LFF Eclat, and then by the remaining algorithms in the same order as before, w.r.t. amounts of candidates generated.

These conclusions are in agreement with previous experimental results [4, 5, 13]. Some of these conclusions could be obtained by considering particular fixed datasets but our probabilistic analysis shows that these conclusions are quite general.

7. REFERENCES

[1] R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. *Proc. ACM SIGMOD Int. Conf. Management of Data*, pages 207–216, 1993.

[2] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. *Proc. of VLDB Conference*, pages 487–499, 1994.

[3] R. Agrawal and R. Srikant. Mining sequential patterns. *Proc. IEEE ICDE Int. Conf. on Data Engineering*, pages 3–14, 1995.

[4] R. J. Bayardo. Efficiently mining long patterns from databases. *Proc. ACM SIDMOD Int. Conf. on Management of Data*, pages 85–93, 1998.

[5] R. J. Bayardo, B. Goethals, and M. J. (co-chairs) Zaki. Workshop on frequent itemset mining implementations (fimi '04). *Brighton, UK*, 2004.

[6] T. Calders. Computational complexity of itemset frequency satisfiability. *Proc. ACM SIGACT-SIGMOD-SIGART Symp. on Principles of Database Systems*, pages 143–154, 2004.

[7] H. Chernoff. A measure of asymptotic efficiency for test of a hypothesis based on the sum of observations. *Annals of Mathematical Statistics*, 23:493–507, 1942.

[8] F. Geerts, B. Goethals, and J. Van den Bussche. A tight upper bound on the number of candidate patterns. *Proc. of the first IEEE Int. Conf. on Data Mining*, 2001.

[9] B. Goethals. Efficient frequent pattern mining. *PhD thesis, transnational University of Limburg, Diepenbeek, Belgium*, December 2002.

[10] J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. *Proc. ACM SIGMOD Int. Conf. Management of Data*, pages 1–12, 2000.

[11] W. A. Kosters and W. Pijls. Apriori, a depth first implementation. *Proc. of the Workshop on Frequent Itemset Mining Implementations*, 2003.

[12] P. W. Purdom, D. Van Gucht, and D. P. Groth. Average case performance of the apriori algorithm. *SIAM J. Computing*, 33 (5):1223–1260, 2004.

[13] M. J. Zaki. Scalable algorithms for association mining. *IEEE Transactions on Knowledge and Data Engineering*, 12 (3):372–390, 2000.

[14] Z. Zheng, R. Kohavi, and L. Mason. Real world performance of association rule algorithms. *Proc. of the 7th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, pages 401–406, 2001.