Estimating the Number of Frequent Itemsets in a Large Database

Ruoming Jin Scott McCallen Yuri Breitbart Dave Fuhry Dong Wang
Department of Computer Science
Kent State University
{jin,smccalle,yuri,dfuhry,dwang}@cs.kent.edu

ABSTRACT

Estimating the number of frequent itemsets for minimal support α in a large dataset is of great interest from both theoretical and practical perspectives. However, finding not only the number of frequent itemsets, but even the number of maximal frequent itemsets, is #P-complete. In this study, we provide a theoretical investigation on the sampling estimator. We discover and prove several fundamental but also rather surprising properties of the sampling estimator. We also propose a novel algorithm to estimate the number of frequent itemsets without using sampling. Our detailed experimental results have shown the accuracy and efficiency of our proposed approach.

1. INTRODUCTION

Given a set of items I and a set of transactions T, each of which is a subset of I, the frequent pattern P (alternatively called a frequent itemset) is defined as a subset of I that occurs in at least $\alpha|T|$ transactions, where α is a number between zero and one and is referred to as the minimum support of pattern P. The problem of finding frequent patterns in a set of given transactions T has been extensively studied [3, 2, 12].

The simple and intuitive concept of frequent itemset mining has found many important applications in business-intelligence environments, web analysis, networking security, and quality control, among others. Recently, a frequent itemset operator has emerged as a new feature supported in commercial databases and data warehouses. This includes Oracle 10g [15], IBM DB2 [22], and SQL Server 9.0 [20].

However, finding frequent itemsets is computationally expensive, especially when the dataset is very large. Furthermore, to discover any meaningful and useful knowledge, the frequent itemset operator may be invoked many times with different parameter values (such as a support level, for example), constraints, and dimensions. Therefore, how to facilitate and support the mining process more effectively, such as reducing the number of executions of the frequent itemset operator, intelligently choosing the right parameters and right dimensions (items), predicting the outcome of the mining results, and even estimating the running time of the frequent itemset operator, is becoming increasingly important for many commercial applications.

All of these desired features of the frequent itemset operator are related to a seemingly simple problem: *what is the number of frequent itemsets for the given minimum support level*? Indeed, if the number of frequent itemsets is large, the data miner may either increase a support level to reduce the number of frequent itemsets or use the number of frequent itemsets to determine an appropriate support level for mining a given dataset. The number of frequent itemsets holds the key for a cost estimation of different data mining algorithms. Given this, predicting the number of frequent itemsets has become a question similar to the traditional database query cardinality estimation problem [8].

The question of efficiently counting the number of frequent itemsets without actually enumerating them, indeed has been a long standing open problem in data mining research. It has been proven [11, 21] that finding the number of frequent itemsets without finding actual frequent itemsets is #P-complete. In other words, efficiently counting the exact number of frequent itemsets for a chosen support level is as hard as enumerating them. Thus, there is little hope that an efficient algorithm for finding the exact number of frequent itemsets will ever be found.

This leads to the central question of this study: "Can we accurately estimate the number of frequent itemsets without actually enumerating them?" In this study, we provide a theoretical investigation on the *sampling estimator*. We discover and prove several fundamental but also rather surprising properties of the *sampling estimator*. We also propose a novel algorithm to estimate the number of frequent itemsets without using sampling. Specifically, our contributions are as follows.

Sampling Estimator: Simply speaking, the sampling estimator tries to estimate the total number of frequent itemsets on the entire dataset by the count of frequent itemsets on a sample dataset. We found this estimator tends to be biased and overestimate the true number significantly for most of the real datasets. We formally prove that it is *asymptotically unbiased* and *consistent* (under certain conditions). We also prove it is biased and derive the specific condition that it can be unbiased. We also provide insight on the overestimating behavior of the sampling estimator.

Besides overestimating, another issue of sampling estimator is its high computational cost. Note that even running the fastest available algorithms on the sample dataset to enumerate all frequent itemsets can still be very expensive due to the large number of frequent itemsets.

Sketch Matrix Estimator Considering these issues of sampling estimator, we ponder the following problem: "Can we construct a *concise* synopsis structure for a transaction database and obtain an estimate only using this synopsis?" In this study, we provide a

EDBT 2009, March 24–26, 2009, Saint Petersburg, Russia.

positive answer to this question.

We propose a novel synopsis, referred to as *sketch* matrix, for the estimation purpose. Simply speaking, we partition the rows (transactions) and columns (items) into M and N disjoint groups, respectively. Thus, the entire transaction dataset is also partitioned into $M \times N$ disjoint parts, i.e., the (i, j) part of the transaction database contains all the transactions in i-th transaction group with the items in the j-th item group. Further, a summary statistics, *density*, is calculate for each part of the database. If we represent the transactional database as a binary matrix, where a cell at k-th row and l-th column with 1 corresponds item i occurring in transaction k and with 0, otherwise, the density of part (i, j) is simply the proportion of 1's in its corresponding submatrix. Given this, the sketch matrix has M rows and N columns, and each cell records the density of the corresponding part in the transaction database. We propose an efficient procedure using the sketch matrix to estimate the number of frequent itemsets. Our sketch matrix construction is inspired by the recent progress in *bi-clustering* research in data mining and machine learning community, which focuses on simultaneous clustering of both rows and columns in a given data matrix [16, 7]. However, the goal here is to construct a sketch matrix which can produce the most accurate approximation. In this study, we propose a new criteria for the bi-clustering in order to minimize the estimation error and develop an efficient algorithm to construct the sketch matrix efficiently.

Estimating other Related Quantities: We note that in many real applications, users are likely to query only the frequent itemsets with respect to a subset of items satisfying certain constraints. Such conditions have been extensively investigated in the area of constraint data mining. Our estimators can easily be applied to such scenarios. Further, our techniques can provide accurate approximation for the number of k-frequent itemsets and the maximal frequent itemset size as well.

Experimental Evaluation: We conduct extensive experiments on the publicly available sets of transactions on both the sampling estimators and the sketch matrix estimators. We found the sketch matrix estimator has a rather constant estimation time and is much faster than the sampling-based estimator on the dense datasets. Our experiments show that the sketch matrix estimator can obtain the approximate number of frequent itemsets within 70%-90% of the exact number for the tested datasets.

2. SAMPLING ESTIMATOR

Let T be the set of all transactions of the entire transactional database D, and I be the set of all items in D. We denote Z to the number of all frequent itemsets on D with respect to the minimal support α: $Z = |\{i : i \subseteq I \wedge f_i \geq \alpha\}|$, where f_i is the true frequency of itemset i in D.

Let S be the sample transaction set which is generated by sampling the transaction set T with replacement of the entire database D. (Our sequel analysis will hold for the sampling without replacement as well). The sampling estimator is denoted as $\hat{Z}$, which counts the total number of frequent itemsets on the sampling dataset S. Let X_i be the number of occurrences of itemset i in S, and Y_i is defined as follows: $Y_i = 1$, if $X_i \geq \alpha|S|$ and $Y_i = 0$, if $X_i < \alpha|S|$. Given this, we can rewrite the sampling estimator as

$$\hat{Z} = \sum_{i \subseteq I} Y_i$$

Clearly, both X_i and Y_i can be treated as random variables. The random variable X_i has binomial distribution (recall we use sampling with replacement):

$$Pr(X_i = l) = \binom{|S|}{l} f_i^l (1 - f_i)^{|S|-l}$$

The random variable Y_i is a Bernoulli trial:

$$Pr(Y_i = 1) = Pr(X_i \geq \alpha|S|) = \sum_{l \geq \alpha|S|} Pr(X_i = l)$$

$$Pr(Y_i = 0) = Pr(X_i < \alpha|S|) = \sum_{l < \alpha|S|} Pr(X_i = l)$$

In the following, we first report some positive results for the sampling estimator (Subsection 2.1) and then we discuss some rather negative properties of the estimator (Subsection 2.2).

2.1 Asymptotic behavior of $\hat{Z}$

Here, we study the properties of $\hat{Z}$ when the sample size become increasingly large.

DEFINITION 1. *[14] Let $E(\hat{\theta}_n)$ be the estimator of parameter θ for the sampling population with n samples. The estimator $\hat{\theta}$ is said to be* asymptotically unbiased *if*

$$\lim_{n \to \infty} E(\hat{\theta_n}) = \theta$$

The estimator $\hat{\theta}$ is said to be consistent *if any fixed $\epsilon > 0$,*

$$\lim_{n \to \infty} Pr(|\hat{\theta}_n - \theta| \geq \epsilon) = 0$$

Essentially, an asymptotically unbiased estimator will converge to the true value of the estimated parameter when the sample size becomes very large (towards infinity). A consistent estimator can produce arbitrarily accurate estimate (the estimate is very close to θ within very small ϵ with probability 1) when the sample size becomes very large (towards infinity). Even though the sample size cannot really become infinity, these two properties do provide good indication on the behavior of the estimator when the sample becomes large. These properties are generally desired for good estimators.

For a minimal support level α, let Z_α be the number of itemsets with exact support α in D: $Z_\alpha = |\{i : f_i = \alpha, i \subseteq I\}|$ and $\overline{Z_\alpha}$ be the number of itemsets with support higher than α in D: $\overline{Z_\alpha} = |\{i : f_i > \alpha, i \subseteq I\}|$. Clearly, the total number for frequent itemsets $Z = Z_\alpha + \overline{Z_\alpha}$. In addition, we denote $Z' = Z_\alpha/2 + \overline{Z_\alpha}$.

THEOREM 1. *The sampling estimator $\hat{Z}$ is asymptotically unbiased for estimating Z'.*

Proof: We first assume no itemset i has support α, and thus $Z = Z'$.

$$\lim_{n \to \infty} E(\hat{Z}_n) = \lim_{n \to \infty} \sum_{i \subseteq I} E(Y_i) = \lim_{n \to \infty} \sum_{i \subseteq I} Pr(X_i \geq \alpha n)$$

$$= \lim_{n \to \infty} \left(\sum_{f_i \geq \alpha} Pr(X_i \geq \alpha n) + \sum_{f_i < \alpha} Pr(X_i \geq \alpha n) \right)$$

$$= \lim_{n \to \infty} \left(\sum_{f_i \geq \alpha} 1 - Pr(X_i < \alpha n) + \sum_{f_i < \alpha} Pr(X_i \geq \alpha n) \right) \quad (1)$$

$$f_i > \alpha : \lim_{n\to\infty} Pr(X_i < \alpha n) = \lim_{n\to\infty} Pr(X_i \le (1-\delta) f_i n)$$

$$\le \lim_{n\to\infty} e^{-\delta^2 f_i n/2} = 0 \; (Chernoff\ Bounds) \quad (2)$$

$$f_i < \alpha : \lim_{n\to\infty} Pr(X_i \ge \alpha n) = \lim_{n\to\infty} Pr(X_i \ge (1+\delta) f_i n)$$

$$\le \lim_{n\to\infty} e^{-\delta^2 f_i n/3} = 0 \; (Chernoff\ Bounds) \quad (3)$$

$$(2)\&(3) \Rightarrow (1) = Z' = Z \quad (4)$$

Now, we consider $Z_\alpha \neq 0$ and we will show $\lim_{n\to\infty} \sum_{f_i=\alpha} E(Y_i) = Z_\alpha/2$.

$$\lim_{n\to\infty} \sum_{f_i=\alpha} E(Y_i) = \lim_{n\to\infty} \sum_{f_i=\alpha} Pr(X_i \ge \alpha n)$$

$$(Central\ Limit\ Theorem, x = X_i/n, \sqrt{n}\frac{(x-\alpha)}{\alpha(1-\alpha)} \sim N(0,1))$$

$$= \lim_{n\to\infty} \sum_{f_i=\alpha} \int_\alpha^\infty \frac{1}{\sqrt{2\pi n\sigma}} e^{-2(x-\alpha)/2n\sigma^2} dx \;\; (\sigma = \alpha(1-\alpha))$$

$$= \frac{1}{2}|\{f_i = \alpha\}| = Z_\alpha/2 \quad (5)$$

$$(4)\&(5) \Rightarrow \hat{Z} = Z'$$

□

An asymptotically unbiased estimator does not necessarily have to be consistent, but a consistent estimator must be at least asymptotically unbiased. Indeed, Theorem 2 shows that the sampling estimator is consistent for Z' when no itemset i has support α. However, when $Z_\alpha \neq 0$, we will show this property does not hold any more.

THEOREM 2. *The sampling estimator $\hat{Z}$ is consistent for estimating Z' when $Z_\alpha = 0$.*

Proof:

$$\lim_{n\to\infty} Pr(|\hat{Z}_n - Z'| \ge \epsilon) = \lim_{n\to\infty} Pr(|\sum_{i\subseteq I} Y_i - Z'| \ge \epsilon)$$

$$\le \lim_{n\to\infty} \left(Pr(\sum_{f_i>\alpha} Y_i - Z' \le -\epsilon) + Pr(\sum_{f_i<\alpha} Y_i \ge \epsilon) \right)$$

$$\le \lim_{n\to\infty} \left(\sum_{f_i>\alpha} Pr(Y_i - 1 \le -\epsilon) + \sum_{f_i<\alpha} Pr(Y_i \ge \epsilon) \right)$$

$$= \lim_{n\to\infty} \sum_{f_i>\alpha} Pr(Y_i - E(Y_i) + E(Y_i) - 1 \le -\epsilon)$$

$$+ \sum_{f_i<\alpha} Pr(Y_i - E(Y_i) + E(Y_i) - 0 \ge \epsilon)$$

$$\le \lim_{n\to\infty} \sum_{f_i>\alpha} Pr(|Y_i - E(Y_i)| \ge |\epsilon + E(Y_i) - 1|)$$

$$+ \sum_{f_i<\alpha} Pr(|Y_i - E(Y_i)| \ge |\epsilon + E(Y_i)|)$$

$$\le \lim_{n\to\infty} \sum_{f_i>\alpha} \frac{E((Y_i - E(Y_i))^2)}{(\epsilon + E(Y_i) - 1)^2} + \sum_{f_i<\alpha} \frac{E((Y_i - E(Y_i))^2)}{(\epsilon + E(Y_i))^2}$$

$$= \lim_{n\to\infty} \sum_{f_i>\alpha} \frac{Pr(X_i < \alpha n)(1 - Pr(X_i < \alpha n))}{(\epsilon + E(Y_i) - 1)^2} +$$

$$\sum_{f_i<\alpha} \frac{Pr(X_i \ge \alpha n)(1 - Pr(X_i \ge \alpha n))}{(\epsilon + E(Y_i))^2}$$

$$= 0 \quad (From\ (2)\ \&\ (3)) \quad (6)$$

□

Now, we consider $Z_\alpha \neq 0$ and show its inconsistency. Surprisingly, we will show that $\lim_{n\to\infty} Pr(|\hat{Z}_n - Z'| \ge \epsilon)$ can be far way from 0.

$$\lim_{n\to\infty} Pr(|\hat{Z}_n - Z'| \ge \epsilon) \;\; (\epsilon \ll 1)$$

$$\ge \lim_{n\to\infty} Pr(|\sum_{f_i\neq\alpha} Y_i - \overline{Z_\alpha}| < \epsilon \wedge |\sum_{f_i=\alpha} Y_i - Z_\alpha/2| \ge 1)$$

$$(Assuming\ the\ above\ two\ events\ being\ independent)$$

$$\approx \lim_{n\to\infty} Pr(|\sum_{f_i\neq\alpha} Y_i - \overline{Z_\alpha}| < \epsilon) \times Pr(|\sum_{f_i=\alpha} Y_i - Z_\alpha/2| \ge 1)$$

$$= \lim_{n\to\infty} Pr(\sum_{f_i=\alpha} Y_i - Z_\alpha/2 \ge 1) + Pr(\sum_{f_i=\alpha} Y_i - Z_\alpha/2 \le -1)$$

$$= 1 - \lim_{n\to\infty} Pr(\sum_{f_i=\alpha} Y_i = Z_\alpha/2) \approx 1 - \binom{Z_\alpha}{Z_\alpha/2} \times (1/2)^{Z_\alpha}$$

For instance, let $Z_\alpha = 10$, then, we will have $1 - \binom{Z_\alpha}{Z_\alpha/2} \times (1/2)^{Z_\alpha} \approx 0.85$.

However, we can show the $\hat{Z}$ is indeed quite close to Z' (within a range of $Z_\alpha/2 + \epsilon$ in probability 1).

THEOREM 3. *When $Z_\alpha \neq 0$, $\lim_{n\to\infty} Pr(|\hat{Z}_n - Z'| \ge Z_\alpha/2 + \epsilon) = 0$.*

Proof: Omitted for simplicity. □

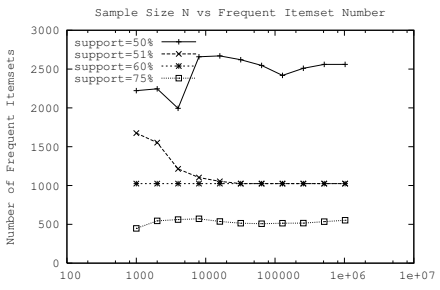
We note that for the sampling without replacement, $\hat{Z}_n$ would share the similar behaviors (not converging to Z) as described in Theorem 1, 2, and 3, as $n \to |T|$ ($n < |T|$).

Observing the Behavior of $\hat{Z}$ in Theorems 1, 2, and 3: To observe the asymptotic behavior of the $\hat{Z}$, we sample the following transactional database with 100 transactions: 50 of them are $\{1,2,3,4,5,6,7,8,9,10,11,12\}$; 25 of them are $\{1,2,3,4,5,6,7,8,9,10\}$, and another 25 are $\{1,2,3,4,5\}$. We perform the random sampling with replacement to generate sample database with number of transactions: 1000, 2000, $\cdots$, $2^{10} \times 1000$. We generate 500 random sample database for each particular number of transactions. Figure 1(a) shows the average number of frequent itemsets for the 500 random sample database at support level 50%, 51%, 60% and 75%, where the exact number of frequent itemsets on the entire database are $2^{12} - 1 = 4095$, and $2^{10} - 1 = 1023$, $2^{10} - 1 = 1023$, and $2^{10} - 1 = 1023$, respectively.

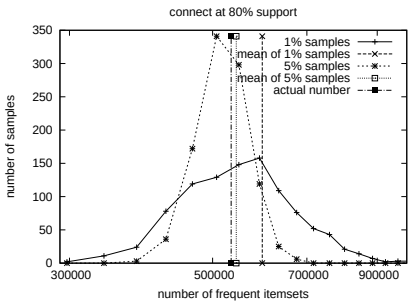
We make the following observations: 1) It is easy to see that the average number of frequent itemsets for support 51% and 60% converge to the exact count. However, as we vary the number of sample size from 1000 to $1,024,000$, the average number of frequent itemsets for 50% and 75% are no where near their exact number. Theorem 1 predicts they will converge to $2^{10}-1+(2^{12}-2^{10})/2 = 2559$ for support 50% and $2^5 - 1 + (2^{10} - 2^5)/2 = 527$ for support 75%, and explains their convergence behavior. 2) When $Z_\alpha = 0$, i.e., at support level 51% and 60%, almost all the sample database would produce the same number of frequent itemsets 1023. However, when $Z_\alpha \neq 0$, e.g., at support level 50%, the sample database would procedure either $2^{12} - 1 = 4095$ or $2^{10} - 1 = 1023$ frequent itemsets. Even though their average is 2559, each individual sample dataset has very different number of frequent itemsets. But they are all within the range predicted by Theorem 3. 3) Another interesting behavior is for support 51%, when the sample size is not very large, the sampling estimator seems always overestimate (upper convergence) the true value. This will be the topic of Subsection 2.2.

2.2 Bias and Variance of $\hat{Z}$

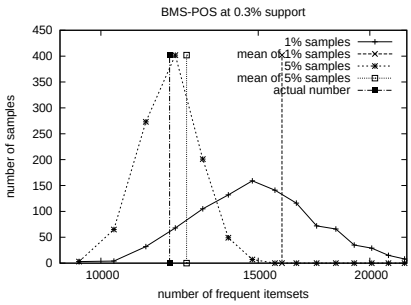
Note that the asymptotic property holds only when the sample is very large. However, we generally will be able to afford a large



(a) synthetic sample



(b) connect sample count



(c) BMS-POS sample count

sample as our estimator will be used for cardinality estimation before the query processing. In the following, we take a detailed look of the bias and the variance of the sampling estimator $\hat{Z}$ assuming the sample is not large. In the meantime, we will focus on estimation Z, instead of Z' (Similar results hold for Z' as well).

DEFINITION 2. *[14] Let $\hat{\mathcal{Z}}$ be an estimator of Z. The bias of estimator $\hat{\mathcal{Z}}$ is defined as*

$$Bias(\hat{\mathcal{Z}}) = E(\hat{\mathcal{Z}}) - Z$$

The estimator $\hat{\mathcal{Z}}$ is unbiased *if the bias is 0, i.e., the expectation of the estimator is equivalent to the true value:*

$$E(\hat{\mathcal{Z}}) = Z$$

First, we show that $\hat{Z}$ is biased, i.e., $\hat{Z} \neq Z$ by a counterexample. Let all the items be split into two sets I_1 and I_2, where $|I_1| = |I_2|$ and $I_1 \cap I_2 = \emptyset$. Let us assume half of the transactions in the database contains I_1 and the other half contains I_2. Let the minimal support $\alpha = 55\%$. Then,

$$\begin{aligned} E(\hat{Z}) &= \sum_{i \subseteq I_1} E(Y_i) + \sum_{i \subseteq I_2} E(Y_i) \\ &= 2 \times 2^{|I|/2} Pr(X_i \geq 55\%n) \neq 0 \end{aligned}$$

Bias Analysis of Sampling Estimator: The bias of the $\hat{Z}$ can be written as

$$\begin{aligned} Bias(\hat{Z}) &= E(\hat{Z}) - Z = \sum_{i \subseteq I} Pr(X_i \geq \alpha n) - Z \\ &= \sum_{f_i \geq \alpha} Pr(X_i \geq \alpha n) + \sum_{f_i < \alpha} Pr(X_i \geq \alpha n) - Z \\ &= \sum_{f_i \geq \alpha} 1 - Pr(X_i < \alpha n) + \sum_{f_i < \alpha} Pr(X_i \geq \alpha n) - Z \\ &= \sum_{f_i < \alpha} Pr(X_i \geq \alpha n) - \sum_{f_i \geq \alpha} Pr(X_i < \alpha n) \end{aligned}$$

Clearly, only when

$$\sum_{f_i < \alpha} Pr(X_i \geq \alpha n) = \sum_{f_i \geq \alpha} Pr(X_i < \alpha n),$$

the estimate will be unbiased, i.e., $Bias(\hat{Z}) = 0$.

Why Sampling Estimator Typically Overestimates Z? In general, the sampling estimator $\hat{Z}$ tends to significantly *overestimate* the true number of frequent itemsets, Z [1]. In other words, $Bias(Z) \gg 0$,

$$\sum_{f_i < \alpha} Pr(X_i \geq \alpha n) \gg \sum_{f_i \geq \alpha} Pr(X_i < \alpha n)$$

For instance, Figures 1(b) and 1(c) show the empirical distribution of the sampling estimator on the public available datasets connect and BMS-POS [1]. Here, we sample the original datasets with replacement for 1000 times at 1% and 5% sampling ratio. The two curves report the number of frequent itemsets versus the number of sample trials. Clearly, in both datasets, the sampling estimator tends to overestimate the actual number of frequent itemsets. Indeed, this is rather counterintuitive and seems contradicting the fact (Theorem 1) that $E(\hat{Z})$ converges to Z', which is smaller than or equal to Z. Why does it behave like this? Here, we perform some simple analysis to reveal the underlying cause. First, for an itemset i, the difference between the estimated frequency $X_i/|S|$ and its true support $f_i|S|$ can be bound by

$$Pr(|X_i/|S| - f_i| > \epsilon) < 2e^{-2\epsilon^2|S|}$$

using Chernoff bounds [18]. Suppose $\epsilon = 1\%$, and $|S| = 10,000$, then the probability for the difference between the estimated frequency and the true frequency is less than 25%. This suggests that if an itemset has frequency f_i being close to the targeted support level, $|f_i - \alpha| < \epsilon$, it is very likely to jump from frequent to infrequent or infrequent to frequent. However, when the frequency of an itemset is either much lower or much higher than the target support level α, the probability of such jump is very small. Given this, we simplify the bias by considering only itemsets whose support is close to the support level, i.e., for certain very small ϵ,

$$\begin{aligned} Bias(Z) &\approx \sum_{\alpha-\epsilon<f_i<\alpha} Pr(X_i \geq \alpha n) - \sum_{\alpha<f_i<\alpha+\epsilon} Pr(X_i < \alpha n) \\ &\approx \sum_{\alpha-\epsilon<f_i<\alpha} \int_{\alpha}^{\infty} e^{-(X_i-f_in)^2/(2f_i(1-f_i)n)^2} - \\ &\quad \sum_{\alpha<f_i<\alpha+\epsilon} \int_{-\infty}^{\alpha} e^{-(X_i-f_in)^2/(2f_i(1-f_i)n)^2} \\ &\approx \sum_{\alpha-\epsilon<f_i<\alpha} 1/2 - \sum_{\alpha<f_i<\alpha+\epsilon} 1/2 \\ &= \frac{1}{2}(|\{i : \alpha - \epsilon < f_i < \alpha\}| - |\{i : \alpha < f_i < \alpha + \epsilon\}|) \end{aligned}$$

The above analysis provides a rule of thumb for estimating the bias of sampling estimator $\hat{Z}$. We observe that for many real datasets, the number of itemsets is shown exponential growth as the support

[1]Note that when $Z_\alpha \neq 0$, the sampling estimator $\hat{Z}$ tends to underestimate Z (unbiased estimator for Z'). However, in most of the cases, Z' will be equal to Z since without prior knowledge, the chance for a user to select a minimal support α, which has $Z_\alpha \neq 0$, is very small. In other words, a typical user-defined minimal support level will not have any itemset with exactly $\alpha|T|$ number of occurrences in the D.

reduces, i.e.,

$$|\{i : \alpha - \epsilon < f_i < \alpha\}| \gg |\{i : \alpha < f_i < \alpha + \epsilon\}|$$

Indeed, if we assume $|\{i : \alpha - \epsilon < f_i < \alpha\}| \approx |\{i : \alpha < f_i\}|$, the bias can be almost as large as the true value Z. This analysis shows that why the observed mean of the sampling estimator can be much larger than Z (e.g., Figures 1(b) and 1(c)).

MSE and Variance Analysis of Sampling Estimator: A biased estimator is not necessarily a bad estimator. Generally, the criteria for a good estimator is the *mean square error* (MSE):

$$MSE(\hat{Z} - Z) = E(\hat{Z} - Z)^2 = Bias(\hat{Z})^2 + Var(\hat{Z})$$

When the bias and variance are both small, a biased estimator can still be desirable. The variance of the sampling estimator can be written as

$$\begin{aligned} Var(\hat{Z}) &= \sum_{i \subseteq I} Var(Y_i) + \sum_{i \neq j} Cov(Y_i, Y_j) \\ &= \sum_{i \subseteq I} nPr(X_i \geq \alpha)(1 - Pr(X_i \geq \alpha) + \\ &\quad \sum_{i \neq j} (Pr(X_i \geq \alpha \wedge X_j \geq \alpha) - Pr(X_i \geq \alpha)Pr(X_j \geq \alpha)) \end{aligned}$$

The direct computation is too expensive. Instead, we can estimate $Var(\hat{Z})$ through re-sampling the original transactional database K times ($\hat{Z}_k$ is the k-th sample dataset):

$$\widehat{Var(\hat{Z})} = \frac{\left(\sum_{k=1}^{K} \hat{Z}_k - \frac{\sum_{k=1}^{K} \hat{Z}_k}{K}\right)^2}{K - 1}$$

An interesting observation based on the empirical distribution of $\hat{Z}$ (e.g. Figures 1(b) and 1(c) is that the sampling estimator does not seem to have normal distribution, but it seems to become normal after we did the logarithm transform on $\hat{Z}$. We conjecture that this may hold for many real datasets. The key open question is under what conditions, this will hold and how to analytically derive it.

3. SKETCH MATRIX ESTIMATOR

In this section, we study how to estimate the total number of frequent itemsets given a sketch matrix $\mathcal{D}$. How to construct such a matrix is discussed in Section 4.

To facilitate our discussion, we introduce the following notations. Let $\mathcal{B}$ be the binary matrix representing the transactional database D. Each row of $\mathcal{B}$ corresponds to a transaction and each column of $\mathcal{B}$ corresponds to an item. b_{ij} in $\mathcal{B}$ is one if and only if the i-th transaction contains item j. Let $\mathcal{D}$ be the sketch matrix with s rows and t columns for the database D. Let $A_i,\ 1 \leq i \leq s$ be the set of transactions being represented by the i-th row of $\mathcal{D}$ and let $a_i = |A_i|$ be the number of transactions for the i-th row. Let $B_j,\ 1 \leq j \leq t$ be the set of items being represented by the j-th column, and let $b_j = |B_j|$ be the number of items for the j-th column. Clearly, we have $\sum_{i=1}^{s} a_i = |T|$ where, $|T|$ is the total number of transactions in D, and $\sum_{j=1}^{t} b_j = |I|$ where, $|I|$ is the total number of items in D. Let the cell at i-th row and j-th column in $\mathcal{D}$, d_{ij}, be the proportion of ones in the submatrix of D, which contains transactions in A_i with only items appearing in B_j. Given a support level α, we would like to estimate the total number of itemsets which have the support higher than or equal to α.

3.1 The Simple Case

Suppose that the sketch matrix $\mathcal{D}$ is a binary matrix where $d_{ij} = 0$ or $d_{ij} = 1$. In this case, it is easy to obtain the exact number of frequent itemsets in a constant time. Indeed, consider the sketch matrix $\mathcal{D}$ depicted in Figure 1.

	b1	b2	b3
a1	1	0	1
a2	0	1	0
a3	1	0	1

a1=a2=a3=100 b1=b2=b3=100

Figure 1: Sketch Matrix Example

Each cell of $\mathcal{D}$ contains 100 transactions and 100 items from the original set of transactions. Suppose that the support level α is 10%. Then it is easy to calculate the precise number of frequent itemsets which is equal to

$$3 * (2^{100} - 1) + (2^{100} - 1)(2^{100} - 1).$$

In general, suppose that k columns $j_1,\ j_2,\ \cdots,\ j_k$ from $\mathcal{D}$ satisfy the condition below.

$$\sum_{i=1}^{s} [d_{ij_1} d_{ij_2} \cdots d_{ij_k}] \times a_i \geq \alpha \times |T|$$

That is, the number of transactions with items from these k columns is at least $\alpha|T|$. We refer to these k columns of $\mathcal{D}$ as frequent k columns. It is well known [3] that any subset of frequent columns is also frequent. Consequently, the number of frequent itemsets for items from frequent k columns of $\mathcal{D}$ is as follows:

$$(2^{b_{j_1}} - 1) \times (2^{b_{j_2}} - 1) \times \cdots \times (2^{b_{j_k}} - 1)$$

Thus, to calculate the number of all frequent itemsets, we calculate the number of frequent itemsets for every combination of frequent item columns and take a sum of these numbers.

3.2 The General Case

Generally, the sketch matrix is not binary. In this subsection we discuss our approach to approximation that is based on probabilistic considerations. Recall that each cell d_{ij} of the sketch matrix $\mathcal{D}$ is derived from the block of original dataset $\mathcal{B}$ (which is a binary matrix of $|T|$ rows and $|I|$ columns). Let $\mathcal{B}(i, j)$ be the submatrix of $\mathcal{B}$ which contains A_i rows and B_j columns. Thus, $\mathcal{B}(i, j)$ is a binary block with a_i rows and b_j columns. Further, $\mathcal{B}(i, j)_{kl} = 1$, $(1 \leq k \leq a_i, 1 \leq l \leq b_j)$ if and only if the k-th transaction of A_i has l-th item of B_j. Given this, we model the number of 1's in each column of the block $\mathcal{B}(i, j)$ as a random variable X_{ij} with a binomial distribution $Bin(n = a_i, p = d_{ij})$, where n is the number of cells in a column of the block and p is the probability that the cell is 1.

Thus, we estimate the number of frequent items for the entire dataset $\mathcal{B}$ by the expected number of frequent items for the entire set of transactions T which is as follows:

$$\sum_{j=1}^{t} b_j Pr(\sum_{i=1}^{s} X_{ij} \geq \alpha|T|).$$

To approximate the number of frequent itemsets resulting from a single column of the sketch matrix we treat the random variables for each column in the same block being independent, and apply the following lemma:

LEMMA 1. *The expected number of frequent itemsets which are*

subsets of B_j (items represented for j-th column) is

$$\sum_{k=1}^{b_j}\binom{b_j}{k}Pr(\sum_{i=1}^{s}X[k]_{ij}\geq\alpha|T|)$$

where, $X[k]_{ij}$ is a random variable with binomial distribution $Bin(a_i,(d_{ij})^k)$.

Given this, we can approximate the total number of frequent itemsets from the entire sketch matrix for a given the minimal support α.

THEOREM 4. *Given the sketch matrix and the binomial distribution assumption for each block, the expected number of all frequent itemsets of the entire dataset is*

$$\sum_{k_1=0}^{b_1}\cdots\sum_{k_t=0}^{b_t}(\binom{b_1}{k_1}\times\cdots\times\binom{b_t}{k_t}\times$$
$$Pr(\sum_{i=1}^{s}X[k_1,\cdots,k_t]_{i[j_1,\cdots,j_t]}\geq\alpha|T|))-1$$

where, $X[k_1,\cdots,k_t]_{i[1,\cdots,t]}$ is a random variable with binomial distribution $\mathcal{B}(a_i,(d_{i1})^{k_1}\times\cdots\times(d_{it})^{k_t})$.

Thus, our approximation for the number of frequent itemsets depends on the probability that the sum of random variables, each one with binomial distribution, is higher than or equal to the minimal support $\alpha|T|$. However, the exact calculation of such a probability is computationally expensive. To avoid computationally expensive probability evaluation, we use a normal distribution $N(u=np,\sigma^2=np(1-p))$ to approximate a binomial distribution $Bin(n,p)$, $np>10$. It is well known that if two random variables are normal and independent, their sum is also normal. Therefore, the sum of binomial random variables can be approximated as

$$\sum_{i=1}^{s}X[k_1,\cdots,k_l]_{i[j_1,\cdots,j_l]}\sim N(\sum_{i=1}^{s}a_id_{ij_1}^{k_1}\times\cdots\times d_{ij_l}^{k_l},$$
$$\sum_{i=1}^{s}a_id_{ij_1}^{k_1}(1-d_{ij_1}^{k_1})\times\cdots\times d_{ij_l}^{k_l}(1-d_{ij_l}^{k_l}))$$

The probability for a normal random variable to be higher than $\alpha|T|$ is easily approximated [6].

Let us analyze the computational complexity to estimate the number of frequent itemsets based on Theorem 4. The inner formula (involving only product) can be computed in a constant time $O(1)$, the time complexity of the entire formula is $O(\prod_{j=1}^{t}(b_j+1))$. This is significantly less than the total search space of enumerating all possible itemsets. For instance, if we have 1000 items in the entire dataset and assume the sketch matrix have 10 columns with each one has 100 items, the complexity for estimation is $100^{10}\approx 2^{70}<<2^{1000}$, where 2^{1000} is the number of all itemsets. On the other hand, this is still too expensive to calculate. To reduce the computational cost, we use a simple heuristic, referred to as *cutoff*:

DEFINITION 3. ***(Cutoff Condition)** Any $k_1'\geq k_1,\cdots,k_t'\geq k_t$, where $\binom{b_1}{k_1}\times\cdots\times\binom{b_t}{k_t}Pr(\sum_{i=1}^{s}X[k_1,\cdots,k_t]_{i[j_1,\cdots,j_t]}\geq\alpha|T|))<1$, will not be counted in estimation of the total number of frequent itemsets.*

Basically, the cutoff heuristic is similar to the apriori principle. Algorithm 1 is the key counting procedure for estimating the total number of frequent itemsets utilizing the cutoff heuristic.

Algorithm 1 $GeneralCount(\mathcal{D},|T|,\alpha)$

```
F ← 1 // number of frequent itemsets from any k columns
S ← [1,1,···,1], |S| = s // support vector
RecursiveCounting(1,S,F) // start counting from first column
Procedure RecursiveCounting(j,S,F)
for l = j to t do
  S' ← S
  F' ← F
  for k = 1 to b_l do
    u ← 0 // Normal Mean
    σ^2 ← 0 // Normal Variance
    for i = 1 to s do
      S'[i] = S'[i] × d_il // a new support vector S'
      u ← u + S'[i] × a_i
      σ^2 ← σ^2 + S'[i] × (1 − S'[i]) × a_i
    end for
    F' ← F' × (b_l − k + 1)/k
    N ← N + F' × Pr(X ≥ α|T|) // r.v.X ∼ N(u,σ^2) //
    Estimated Total Number of FIM
    if F' × Pr(X ≥ α|T|) ≥ 1 then
      RecursiveCounting(l + 1, S', F')
    else
      break
    end if
  end for
end for
```

3.3 Estimating Other Related Quantities

Number of Frequent k-itemsets: An approximation of the number of frequent itemsets each of which contains exactly k items (referred to as frequent k itemset) is a special case of the approximation of the total number of frequent itemsets. Note that in the counting process, we essentially estimate the number of frequent itemsets from different combinations of items of B_j, $1\leq j\leq t$ and summarize each of these cases. Thus, we have the following theorem to estimate the number of frequent k-item sets.

THEOREM 5. *Given the sketch matrix and the binomial distribution assumption for each block, the expected number of frequent k-itemsets of the entire dataset is*

$$\sum_{0\leq k_1\leq b_1,\cdots,0\leq k_t\leq b_t}^{k_1+\cdots+k_t=k}(\binom{b_1}{k_1}\times\cdots\times\binom{b_t}{k_t}\times$$
$$Pr(\sum_{i=1}^{s}X[k_1,\cdots,k_t]_{i[j_1,\cdots,j_t]}\geq\alpha|T|))-1$$

where, $X[k_1,\cdots,k_t]_{i[1,\cdots,t]}$ is a random variable with binomial distribution $\mathcal{B}(a_i,d_{i1}^{k_1}\times\cdots\times d_{it}^{k_t})$.

Clearly, a recursive procedure similar to Algorithm 1 can enumerate all the different combinations of frequent k-itemsets.

Size of the Largest Frequent Itemsets: For the largest frequent itemsets, we simply use the cutoff condition. Mathematically, we estimate the largest frequent K-itemsets to be

$$K=max\{k_1+\cdots+k_t|1\leq k_1\leq b_1,\cdots,1\leq k_t\leq b_t,$$
$$\binom{b_1}{k_1}\times\cdots\times\binom{b_t}{k_t}Pr(\sum_{i=1}^{s}X[k_1,\cdots,k_t]_{i[j_1,\cdots,j_t]}\geq\alpha|T|))\geq 1\}$$

Number of Frequent Itemsets on a Subset of Items: As we mentioned before, in many real applications, users are likely to query only the frequent itemsets with respect to a subset of items satisfying certain constraints. Let I_s be the subset of items which users

are interested in. Given the minimal support level α, we would like to estimate the number of frequent itemsets which are subsets of I_s. The sketch matrix approach can handle this case by simply adjusting the groups of items $B_1, \cdots, B_t$ to $B_1 \cap I_s, \cdots, B_t \cap I_s$, respectively.

THEOREM 6. *Given the sketch matrix and the binomial distribution assumption for each block, the expected number of all frequent itemsets of the entire dataset is*

$$\sum_{k_1=0}^{b'_1} \cdots \sum_{k_t=0}^{b'_t} (({}^{b'_1}_{k_1}) \times \cdots \times {}^{b'_t}_{k_t}) \times$$
$$Pr(\sum_{i=1}^{s} X[k_1, \cdots, k_t]_{i[j_1,\cdots,j_t]} \geq \alpha|T|)) - 1$$

where, $b'_j = |B_j \cap I_s|$, and $X[k_1, \cdots, k_t]_{i[1,\cdots,t]}$ is a random variable with binomial distribution $\mathcal{B}(a_i, (d_{i1})^{k_1} \times \cdots \times (d_{it})^{k_t})$.

In addition, the sketch matrix can be adjusted to handle only part of the transaction set as well. Essentially, we need attach a selection estimator for each transaction group, i.e., to estimate $|A_i \cap T_s|$, where T_s is the subset of transactions. In general, T_s is likely to be expressed as a predict clause. Thus, we only need to adjust each a_i to a'_i in estimating the number of frequent itemsets. Note that this is a little more complicated than estimating the number of frequent itemsets on a set of items since the number of items is generally much smaller than the number of transactions. For item groups, we can explicitly record each item group B_i and perform the intersect operation. However, this (especially the direct intersection) can be too expensive for the transaction group. Thus, we can apply the typical selection estimators [8], which has been extensively studied in relational database research, for such a purpose.

The sketch matrix can naturally adapt to the dynamic environments, where insertions and deletion of transactions are likely to occur. As we will discuss in the next section, the sketch matrix is constructed in an incremental fashion and thus, can adjust to the change easily.

Finally, we note that these quantities can be easily estimated by the sampling estimators as well. However, as we will show later, sampling estimator is in general too computationally expensive to be applied for query cost estimation.

4. OPTIMAL SKETCH MATRIX CONSTRUCTION

The sketch matrix determines the estimation accuracy. Different sketch matrices can provide very different estimation results. The key problem is what is a good criterion for the sketch matrix and how to construct such a matrix. We will answer these two questions in this section.

4.1 Optimal Criterion

Before introducing the optimal criterion, we first need to consider the properties for the sketch matrix, which are essential for the estimator. Specifically, the random variables of each column in a same block $\mathcal{B}(i, j)$ are i.i.d. (independent and identically distributed) according to the binomial distribution $Bin(n = a_i, p = d_{ij})$; all the random variables of the columns in the same row group but in the different column groups, $(X_{i1}, \cdots, X_{it})$, are independent; and the random variables of the columns in the same column $(X_{1j}, \cdots, X_{sj})$ are independent as well. Indeed, if these conditions are satisfied, our sketch matrix estimator can be proved to be *unbiased* for Z' (the sum of the number of frequent itemsets whose support higher than minimal support α and half of the number of the frequent itemsets whose support equals to α). In some sense, the sketch matrix tries to describe the underlying distribution of the transactional database and then directly compute the expected number of frequent itemsets for such a distribution.

However, producing a sketch matrix of D, which satisfy all these conditions, is not easy. First, any statistic tests will not be able to confirm whether these conditions hold. Instead, they will reject the alternative hypothesis, e.g., the dependence assumption. Further, the chi-square independence test for k random variable requires a k-dimensional contingency table and a total of 2^k cells [5]. This is too computationally expensive. Finally, our contingency table is very sparse (including many zeros or very small number of counts in the cells). Even though there are some recent development for the sparse contingency table [13], they would not be able the handle the test at such a scale.

Under such constraints, we proceed with the assumption that independence holds for the random variables and try to directly minimize the variance of the estimator. The experimental results in Section 5 do indicate that such treatment seems to be appropriate and can produce rather accurate estimation.

Here, the variance for the number of frequent itemsets composed of $k_1, \cdots, k_t$ items from $B_{j_1}, \cdots, B_{j_t}$, respectively, can be written as

$$({}^{b_1}_{k_1}) \times \cdots \times ({}^{b_t}_{k_t}) \times Pr[k_1, \cdots, k_t]_{i[j_1,\cdots,j_t]} \times$$
$$(1 - Pr[k_1, \cdots, k_t]_{i[j_1,\cdots,j_t]})$$

where $Pr[k_1, \cdots, k_t]_{i[j_1,\cdots,j_t]} = Pr(\sum_{i=1}^{s} X[k_1, \cdots, k_t]_{i[j_1,\cdots,j_t]} \geq \alpha|T|)$ is a function of $d_{ij}, 1 \leq j \leq t$. Here, we basically treat the events that itemsets being frequent or not as independent Bernoulli trials. Each of them has $Pr[k_1, \cdots, k_t]_{i[j_1,\cdots,j_t]}$ probability being frequent. Thus, the total number of these $(k_1 + \cdots + k_t)$-itemsets can modeled as a random variable with Binomial distribution. Essentially, the smaller the variance of this random variable, the more precise we have our estimation, which is the expectation of this random variable. Unfortunately, though the variance maybe estimated/approximated, the closed analytic form is very hard to derive.

To deal with this problem, we introduce an alternative variance which is closely related to the original variance but much easier to compute. Minimizing the alternative variance results into a suboptimal value of the former variance. Specifically, the alternative variance is defined as follows.

$$Var(\sum_{k_1=0}^{b_1} \cdots \sum_{k_t=0}^{b_t} ({}^{b_1}_{k_1}) \times \cdots \times ({}^{b_t}_{k_t}) X[k_1, \cdots, k_t]_{i[j_1,\cdots,j_t]})$$

The above variance is denoted as $\mathcal{V}$. Here, we basically treat the sum of all the support for every possible itemset in the database as a random variable. Note that in our probabilistic framework, each support is treated as an sum of random variable with binomial distributions (Subsection 3.2). The closed formula of the alternative variance is stated in Theorem 7.

THEOREM 7.

$$\begin{aligned}\mathcal{V} &= Var(\sum_{k_1=0}^{b_1} \cdots \sum_{k_t=0}^{b_t} ({}^{b_1}_{k_1}) \times \cdots \times ({}^{b_t}_{k_t}) X[k_1, \cdots, k_t]_{i[j_1,\cdots,j_t]}) \\ &= \sum_{i=1}^{s} a_i (\prod_{j=1}^{t} (1 + 3d_{ij})^{b_j} - \prod_{j=1}^{t} (1 + d_{ij})^{2b_j})\end{aligned}$$

Proof:

$$\sum_{k_1=0}^{b_1}\cdots\sum_{k_t=0}^{b_t}\binom{b_1}{k_1}\times\cdots\times\binom{b_t}{k_t}X[k_1,\cdots,k_t]_{i[j_1,\cdots,j_t]}$$

$$= \sum_{i=1}^{s}\sum_{q=1}^{a_i}\prod_{j=1}^{t}[\prod_{r=1}^{b_j}(1+x[i,j]_{qr})]$$

$$where,\ x[i,j]_{qr}\sim Bernoulli(d_{ij})$$

$$Thus,\ \mathcal{V}=Var(\sum_{i=1}^{s}\sum_{q=1}^{a_i}\prod_{j=1}^{t}[\prod_{r=1}^{b_j}(1+x[i,j]_{qr})])$$

$$= \sum_{i=1}^{s}\sum_{q=1}^{a_i}Var(\prod_{j=1}^{t}[\prod_{r=1}^{b_j}(1+x[i,j]_{qr})])$$

$$= \sum_{i=1}^{s}\sum_{q=1}^{a_i}(E(\prod_{j=1}^{t}[\prod_{r=1}^{b_j}(1+x[i,j]_{qr})^2])$$

$$-E^2(\prod_{j=1}^{t}[\prod_{r=1}^{b_j}(1+x[i,j]_{qr})])$$

$$= \sum_{i=1}^{s}\sum_{q=1}^{a_i}(\prod_{j=1}^{t}[\prod_{r=1}^{b_j}E((1+x[i,j]_{qr})^2)])$$

$$-(\prod_{j=1}^{t}[\prod_{r=1}^{b_j}E^2(1+x[i,j]_{qr})])$$

$$= \sum_{i=1}^{s}\sum_{q=1}^{a_i}(\prod_{j=1}^{t}(1+3d_{ij})^{b_j})-(\prod_{j=1}^{t}(1+d_{ij})^{2b_j}))$$

$$= \sum_{i=1}^{s}a_i(\prod_{j=1}^{t}(1+3d_{ij})^{b_j}-\prod_{j=1}^{t}(1+d_{ij})^{2b_j})$$

□

4.2 Bi-Clustering Algorithm

Finding the exact sketch matrix which minimizes the variance (objective function) is very hard. Thus, we resort to heuristic algorithm which perform a k-means type bi-clustering [16, 7] to identify the sketch matrix with local minima. The *BiClustering* algorithm accepts user-defined number of rows s and number of columns t and proceeds as follows.

Step 1: (Random Partition) Randomly partition the original dataset into s subsets of transactions and t subsets of items;

Step 2: (Transaction Adjustment) For each transaction, move it to a new group so that the objective function is maximally reduced;

Step 3: (Item Adjustment) For each item, move it to a new group so that the objective function is maximally reduced;

Step 4: (Iteration) Perform step 2 and 3 alternatively until the certain stop condition is satistified, i.e., either a local minimum is reached, or the improvement is too small.

The *BiClustering* algorithm is sketched in Algorithm 2. A major challenge here is that each move needs to recalculate the variance ($\mathcal{V}$), which costs $O(st)$ to compute it from scratch. However, as we try to adjust a transaction or an item, we only need to compute the different between the original variance and the new variance (after the movement). Based on a simple analysis of the variance formula, our algorithm can reduce the cost of the variance difference for a transaction moving to an alternative group to $O(t)$.

The correctness of our algorithm can be derived from Lemma 2.

LEMMA 2. *Let $\mathcal{V}$ be the variance for the current grouping of dataset T. Let $\mathcal{V}'$ be the variance for the new grouping if we*

Algorithm 2 $BiClustering(T, s, t)$

Parameter: The transaction database T
Parameter: The number of transaction (row) group s
Parameter: the number of item (column) group t
// Step 1:
Randomly partition the transactions into s groups
Randomly partition the items into t groups
// Step 2:
$\forall i, 1\le i\le s, V_i \leftarrow a_i(\prod_{j=1}^{t}(1+3d_{ij})^{b_j}-\prod_{j=1}^{t}(1+d_{ij})^{2b_j})$
for each transaction x **do**
  i is the current group transaction x belongs to
  $\forall j, 1\le j\le t, d'_{ij} \leftarrow \frac{a_i\times b_j\times d_{ij}-q_j}{(a_i-1)\times b_j}$ // q_j is the number of items in item-group j of transaction x
  $V'_i \leftarrow a_i(\prod_{j=1}^{t}(1+3d'_{ij})^{b_j}-\prod_{j=1}^{t}(1+d'_{ij})^{2b_j})$
  for each group k, $k\neq i$ **do**
    $V_k \leftarrow a_k(\prod_{j=1}^{t}(1+3d_{kj})^{b_j}-\prod_{j=1}^{t}(1+d_{kj})^{2b_j})$
    $\forall j, 1\le j\le t, d'_{kj} \leftarrow \frac{a_k\times b_j\times d_{ij}-q_j}{(a_k-1)\times b_j}$
    $V'_k \leftarrow a_k(\prod_{j=1}^{t}(1+3d'_{kj})^{b_j}-\prod_{j=1}^{t}(1+d'_{kj})^{2b_j})$
    $\Delta_k \leftarrow V_i+V_k-V'_i-V'_k$
  end for
  $k=max(\Delta_k), \Delta_i=0$ // moving x from i-th group to k-th group maximally reduce the variance
  if $i\neq k$ **then**
    $A[i]\leftarrow A[i]-\{x\}$, $A[k]\leftarrow A[k]\cup\{x\}$
    $\forall j, 1\le j\le t, d_{ij}\leftarrow d'_{ij}, d'_{kj}$
  end if
end for
// Step 3:
$\forall i, 1\le i\le s, p_i=\prod_{j=1}^{t}(1+3d_{ij})^{b_j}, p'_i=\prod_{j=1}^{t}(1+d_{ij})^{2b_j}$
for each item y **do**
  j is the current group item y belongs to
  $\forall i, 1\le i\le s, d'_{ij} \leftarrow \frac{a_i\times b_j\times d_{ij}-q_i}{a_i\times(b_j-1)}$ // q_i is the number of transactions in transaction group i containing item y
  for each group l, $l\neq j$ **do**
    $\forall i, 1\le i\le s, d'_{il} \leftarrow \frac{a_i\times b_l\times d_{il}+q_i}{a_i\times(b_l-1)}$
    $\Delta_l \leftarrow \sum_{i=1}^{s}a_i[p_i\times\frac{(1+3d'_{ij})^{b_j-1}}{(1+3d_{ij})^{b_j}}\times\frac{(1+3d'_{il})^{b_l+1}}{(1+3d_{il})^{b_l}}-$
      $p'_i\times\frac{(1+3d'_{ij})^{2(b_j-1)}}{(1+3d_{ij})^{2b_j}}\times\frac{(1+3d'_{il})^{2(b_l+1)}}{(1+3d_{il})^{2b_l}}]$
  end for
  $l=max(\Delta_l), \Delta_j=0$
  // moving y from j-th group to l-th group maximally reduce the variance
  if $j\neq l$ **then**
    $B[j]\leftarrow B[j]-\{y\}, B[l]\leftarrow B[l]\cup\{y\}$
    $\forall i, 1\le i\le s: d_{ij}\leftarrow d'_{ij}, d_{il}\leftarrow d'_{il}$,
    $p_i \leftarrow p_i\times\frac{(1+3d'_{ij})^{b_j-1}}{(1+3d_{ij})^{b_j}}\times\frac{(1+3d'_{il})^{b_l+1}}{(1+3d_{il})^{b_l}}$
    $p'_i \leftarrow p'_i\times\frac{(1+3d'_{ij})^{2(b_j-1)}}{(1+3d_{ij})^{2b_j}}\times\frac{(1+3d'_{il})^{2(b_l+1)}}{(1+3d_{il})^{2b_l}}$
  end if
end for
// Step 4:
Repeat Step 2 and 3 until the stop condition is satistied

move transaction x from its original group i, A_i to a new group $k, A_k, i \neq k$. Let $\mathcal{V}''$ be the variance for the new grouping if we move item y from its original grouping j, B_j to a new grouping $l, B_l, j \neq l$. Then, we have

$$\mathcal{V} - \mathcal{V}' = \Delta_k \ \ and \ \ \mathcal{V} - \mathcal{V}'' = \Delta_l$$

Further, if $\Delta_k > 0$, then $\mathcal{V} > \mathcal{V}'$; and if $\Delta_l > 0$, then $\mathcal{V} > \mathcal{V}''$.

Note that Δ_k is defined in Line 12 of the Algorithm 2 and Δ_l is defined in Line 27 of the Algorithm 2. If we choose row group k and column group l such that they maximally reduce the variance $\mathcal{V}$, we simply choose the maximal Δ_k and Δ_l for a transaction and an item, respectively. We also note that this lemma allows to have many different moves to adjust the grouping so that we can minimize the variance $\mathcal{V}$. In the *BiClustering* algorithm we adjust rows and then adjust columns. Since for each adjustment, we do not increase the variance, the algorithm will eventually converge to a local minimum.

The time complexity of the *BiClustering* algorithm is as follows. In Step 2, we adjust each transaction. The cost for calculating Δ_k for each alternative group is $O(t)$. There are $s-1$ alternative groups. Putting all these together, it costs $O(st)$ to adjust one single transaction. Thus, the total cost of Step 2 is $O(|T|st)$. Similarly, we have the total cost for Step 3 to be $O(|I|st)$. Therefore, assuming the algorithm will iterate L times and the initial density calculation for the entire dataset, then the cost of the entire algorithm is $O(N + L(|T| + |I|)st)$, where N is the size of the entire dataset T.

Finally, we note that the cost of creating a sketch matrix is amortized for obtaining approximation of the number of frequent itemsets for different minimal support levels. In addition, since this algorithm is inherently executed in an incremental fashion, it can quickly adjust for database changes without starting from scratch.

4.3 Two-level Hierarchical Bi-clustering

A key advantage of sketch matrix is its computational cost. However, this cost is determined by the size of the sketch matrix. As the number of rows and columns of a sketch matrix increases, the estimation cost will increase as well (Algorithm 1). However, a small sketch matrix may not be able to capture the underlying distribution of a transactional database very well, in the sense, that the independence assumption may not hold and the variance is large, consequently, the estimation is rather inaccurate.

To handle such a difficulty, we propose a two-level hierarchical clustering method to improve the estimation accuracy with minimal increasing of the estimation cost. The basic idea of the hierarchical clustering is as follows. At the first level, we apply the *BiClustering* algorithm to partition all transactions into s row groups, and all items into t item groups. Thus, we have a total of $s \times t$ blocks. At the second level, we explore the *local structure* of each block by partitioning them further into $s' \times t'$ blocks. That means the entire dataset T is partitioned into a total of $s \times t \times s' \times t'$ blocks. However, as we will show later, the actual estimation does not treat this sketch matrix as $s \times t \times s' \times t'$.

To achieve this, we enforce the following constraints for the second level clustering: *all blocks in the same column generated by the first level clustering shares the same column clustering (item grouping) in the second level clustering, but no constraints for the second level transaction grouping.* Consider for a column group B_j in the first level. This constraint essentially would split it into t' subgroups. However, for a row group A_i, different blocks sharing A_i may split it very differently at the second level.

Construction Procedure: The current *BiClustering* algorithm can be easily modified to handle the second level clustering. Basically, after the first level clustering, we assemble all the blocks in the same column as one sub-dataset. Thus we have a total of t sub-datasets, which are denoted as $T_1, \cdots, T_t$. We then cluster each of them with the following constraints: all the items in the sub-dataset will be partitioned into t' groups, and each existing row group based on the first level clustering will be partitioned into s' groups. In the transaction adjustment process, each transaction is allowed to switch to the subgroups belonging to its original group. The time complexity for the second level clustering is $O(N + L'(|T| \times t'ts' + |I| \times ss't'))$, assuming each sub-dataset converge in L' iterations.

Estimation Procedure: The estimation procedure utilizing the two-level sketch matrix splits the estimation into two steps: 1) using the sketch matrix generated by the second level clustering to estimate the number of frequent itemsets for each sub-dataset T_j. Each sub-dataset T_j is treated as the entire dataset and the thus procedure in Algorithm 1 can estimate the number of frequent itemsets from the each sub-dataset; 2) estimating the number of frequent itemsets which combines itemsets from more than one sub-datasets.

The key trick then is that for each different type of itemsets (each type correspond to a fix number of items from each item-group at the second level clsutering), their support in each block (generated in the first level clustering) is recorded. For instance, consider item group A_i is split into three subgroups, A_{i0}, A_{i1}, and A_{i2}. Let us denote $A_i[2,3,1]$ to be an itemset type which has 2 items in A_{i0}, 3 items in A_{i1} and 1 item in A_{i2}. After calculating its support in using the second level clustering. We will compute a new density for this itemset at each block (in the first level clsutering), denoted as $d_{ij}[2,3,1]$ for the j-th column block. Note that this density can be different from the original d_{ij}, which records the density for the $\mathcal{B}(i,j)$ block. Here, this new density varies from one itemset type to another type, and is computed based on the second level clustering. In particular, this density will be recorded for each itemset type, and each type has s new density values corresponding to the s transaction groups in the first level of clustering. Given this, we can easily combine different types of frequent itemsets from different column groups and use their new densities to do the estimation. Finally, we note that this procedure is efficient since it estimates the number of frequent itemsets for the second step using only t transactional groups instead of $t \times t'$.

5. EXPERIMENTAL EVALUATION

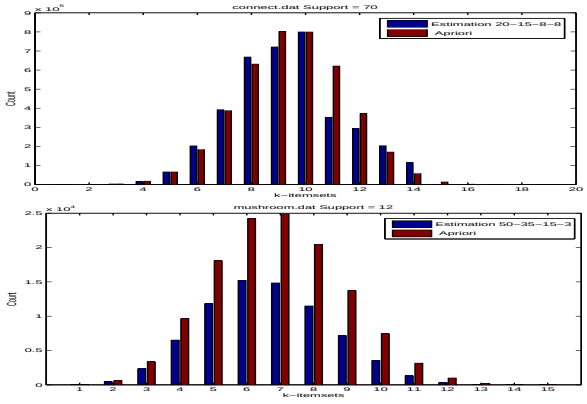


Figure 2: Estimation of the Number of Frequent K-Itemsets

In the experiments, we use five publicly available datasets from the Frequent Itemset Mining Implementations (FIMI) Repository [1].

The datasets are: *accidents*, *chess*, *connect*, *mushroom*, and *retail* [4, 10]. The characteristics of these datasets are listed in Table 5. Datasets $accidents$, $chess$, and $connect$ are rather dense (meaning a large number of frequent itemsets exists at a high support level, for instance 90%). $Retail$ dataset is very sparse (frequent itemsets are generated only at a very low support level, ranging from 0.01% to 0.25%). The mushroom dataset is moderately dense with a range of tested support levels between $8 - 40\%$.

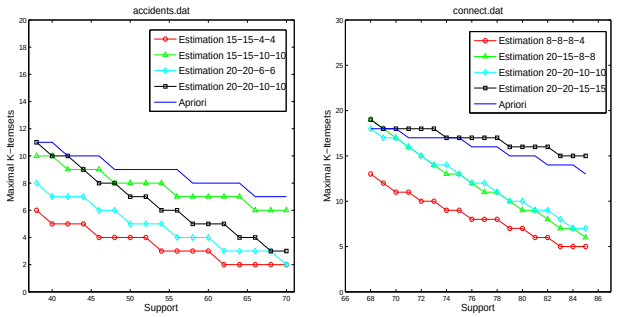
Dataset	Transactions	Items	Sparsity
accidents.dat	340183	468	Dense
chess.dat	3196	75	Dense
connect.dat	67557	129	Dense
mushroom.dat	8124	119	Moderate
retail.dat	88126	16470	Very Sparse

Our experiments are performed on a computation server equipped with Dual AMD Opteron 270 Dual Core Processors and 2.0 GB of main memory. The operating system is the Fedora Core Linux. All algorithms were implemented in C++.

For each dataset the approximation is performed on different sketch matrix configurations. Each configuration is specified by four parameters: 1) s is the number of transaction groups, 2) t is the number of item groups, 3) j is the number of sub transaction groups and 4) k is the number of sub item groups. We denote each configuration as s-t-j-k. In the following, we report the approximation accuracy and running time for different datasets and different for configurations.

Estimators Comparison: Sampling Estimator vs. Sketch Matrix Estimator:. Figures 4 and 5 compare the the sampling estimator with sketch matrix estimator on the connect and mushroom dataset, respectively. Here, we sample the original transactional dataset without replacement at 0.5%, 1% and 3% ratio, and then we apply the state-of-art LCM [19] (one of fastest software) for enumerating the number of frequent itemsets. To show the distribution of the sampling estimator, we generate 100 sample datasets at each sampling ratio.

Figure 4(a) and (b) show the estimation results from the sampling estimator and the sketch matrix estimator with two different configurations, $20 - 15 - 8 - 8$ and $20 - 20 - 10 - 10$ for connect. Clearly, the sketch matrix provides much accurate estimation than the sampling estimator estimator. Figure 4(c) and (d) show the average running time for the sampling estimator (the LCM running time on the sample dataset) and approximation time for the sketch matrix estimator. Interestingly, as sample size increases, the average running time also reduces slightly. This is well captured by our theoretical analysis of the sampling estimator: when the sample is small, the sampling estimator tends to overestimate the number of frequent itemsets and when the sample becomes large, such overestimation reduces, and thus the running time actually can reduce.



(a) (b)

Figure 3: Estimation of the Size of Largest Itemsets

Figure 5(a) and (b) show the estimation results from the sampling estimator and the sketch matrix estimator with two different configurations, $35 - 20 - 10 - 10$ and $50 - 35 - 15 - 3$ for mushroom. Here, the two methods seem comparable. The sampling estimator has probability around 85% and 75% and 60% to overestimate the true number of frequent itemsets for the sampling ratio 0.5%, 1% and 3%, respectively. The sketch matrix tends to underestimate the true number. Figure 5(c) and (d) show the running time comparison. In this case, the sampling estimator actually runs much faster than the sketch matrix. We illustrate this figure to show that the sampling estimator can be acceptable when the dataset is not very dense and the number of itemsets is not very large. Here, the number of frequent itemsets is less than 1000 at both support level. We also note that even though the sketch matrix is slower than the sampling estimator, its running time is still acceptable (less than 0.4 seconds).

In general, sketch matrix estimator can be completed within 1 second (or much less). For instance, it completes all the estimation for retail in averaging 0.003 seconds and accidents averaging 0.008 seconds. Compared with 1% sample for connect and accidents, sketch matrix is more than 1000 and 50 times faster, respectively. To sum, the sketch matrix is more accurate at lower support levels and also for the dense datasets making it especially applicable for cardinality approximation. The sampling estimator can be applied for sparse datasets and generally with higher support level.

In the following, we will mainly focus on studying the sketch matrix estimator.

Estimation of the total number of frequent itemsets:. Figures 6 reports the approximation accuracy for the total number of frequent itemsets. Here, we vary the support level and report both the true count our approximation based on four sketch matrix configurations.

We have the following observations. First, we found that the finer the partitions (i.e. higher s and t), the better the approximation of the true count of frequent itemsets. This is understandable as the finer the partition, the more precise summarization can be achieved for the underlying dataset. Second, the approximations for the dense datasets, chess, connect, mushroom and accidents, are very accurate. If we define the accuracy as E/T, where E is the approximation and T is the true count, the best approximation of these datasets is consistently within or close to 80%. Third, while the approximation for the sparse (retail) dataset is not as precise, it still provides reasonably good approximations at 70% accurracy. Fourth, we can observe that the approximation algorithm generally underestimates all true counts, thus, providing a lower bound for the total number of itemsets for a given support level. This phenomena can be partially explained by the assumption that each of the items is an independent random variable with probability equal to the density of the block it is in.

Estimation of the number of frequent k-itemsets:. Figure 2 shows the detailed approximation of the number of k-itemsets. We can see overall, our approximation algorithm not only provides a good accuracy for the total number of frequent itemsets, but also can estimate each component (k-itemsets) reasonably well. Figure 2(a) shows the approximation of k-itemsets for the connect dataset at support level 70%. Figure 2 (b) is the result for the mushroom dataset at support 12%.

Estimating the large K for the frequent K-itemsets:. Figure 3 (a) and (b) show the approximation of the size of the largest frequent itemsets denoted as K. Our approximation of K for the connect dataset with a $20-20-15-15$ configuration and for the accidents dataset with a $15-15-10-10$ configuration are both very accurate.

Finally, we note that memory cost for the sketch matrix is also very small which is related to the sketch matrix size which is bounded by $s \times t \times j \times k$. In the experiments, the largest sketch matrix was 78.2KB (retail.dat) which is much smaller than the original dataset.

6. RELATED WORK

Finding the number of frequent itemsets and number of maximal frequent itemsets has been shown to be #P-complete [11, 21]. So far, little work has been done to address the problem of estimating the number of frequent itemsets. Implicitly the problem of finding the number of frequent itemsets was addressed in [9], where the authors provided the estimate for the number of frequent itemset candidates containing k elements. However, since the set of candidate frequent itemsets can be much larger than the true frequent itemsets, this method cannot serve as a precise cardinality estimation for the frequent itemsets. In [17] the authors theoretically estimate the average number of frequent itemsets under the assumption that the matrix B, representing the set of transactions, is subject to either simple Bernoulli or Markovian model. In contrast, our approach does not make any probabilistic assumptions about the set of transactions. We design efficient algorithms to build the sketch matrix which effectively summarizes the transaction database and then we estimate the number of frequent itemsets using this matrix.

7. CONCLUSIONS AND FUTURE WORK

In this paper we argue that estimating the number of frequent itemsets is an important problem in data mining. Knowing this number allows us to regulate the computational complexity of the generation of frequent itemsets at different minimal support levels. We perform a detailed study on sampling estimator and propose a new sketch matrix estimator. Overall, we hope that the results obtained here will start a promising direction in an optimization of data mining techniques that deal with the frequent itemsets generation. Our goal is to build a robust and fast cardinality estimation engine for the frequent itemset mining operator.

8. REFERENCES

[1] Frequent itemset mining data repository. http://fimi.cs.helsinki.fi/data.

[2] R. Agrawal, H. Mannila, R. Srikant, H. Toivonent, and A. Inkeri Verkamo. Fast discovery of association rules. In U. Fayyad and et al, editors, *Advances in Knowledge Discovery and Data Mining*, pages 307–328. AAAI Press, Menlo Park, CA, 1996.

[3] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proc. of Int. conf. Very Large DataBases (VLDB'94)*, pages 487–499, Santiago,Chile, September 1994.

[4] Tom Brijs, Gilbert Swinnen, Koen Vanhoof, and Geert Wets. Using association rules for product assortment decisions: a case study. In *KDD '99: Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 254–260, 1999.

[5] S. Brin, R. Motwani, and C. Silverstein. Beyond market baskets: Generalizing association rules to correlations. In *ACM SIGMOD Conf. Management of Data*, May 1997.

[6] George Casella and Roger L. Berger. *Statistical Inference, 2nd. Edition*. DUXBURY Publishers, 2001.

[7] Deepayan Chakrabarti, Spiros Papadimitriou, Dharmendra S. Modha, and Christos Faloutsos. Fully automatic cross-associations. In *KDD '04: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 79–88, 2004.

[8] Surajit Chaudhuri. An overview of query optimization in relational systems. In *PODS Conference Proceedings*, pages 34–43, 1998.

[9] Floris Geerts, Bart Goethals, and Jan Van Den Bussche. Tight upper bounds on the number of candidate patterns. *ACM Trans. Database Syst.*, 30(2):333–363, 2005.

[10] Karolien Geurts, Geert Wets, Tom Brijs, and Koen Vanhoof. Profiling high frequency accident locations using association rules. In *Proceedings of the 82d Annual Transportation Research Board*, 2003.

[11] Dimitrios Gunopulos, Roni Khardon, Heikki Mannila, Sanjeev Saluja, Hannu Toivonen, and Ram Sewak Sharma. Discovering all most specific sentences. *ACM Trans. Database Syst.*, 28(2), 2003.

[12] J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. In *Proceedings of the ACM SIGMOD Conference on Management of Data*, 2000.

[13] Donguk Kim and Alan Agresti. Nearly exact tests of conditional independence and marginal homogeneity for sparse contingency tables. *Comput. Stat. Data Anal.*, 24(1):89–104, 1997.

[14] E. L. Lehmann and George Casella. *Theory of Point Estimation, 2nd Edition*. Springer-Verlag, 1998.

[15] Wei Li and Ari Mozes. Computing frequent itemsets inside oracle 10g. In *VLDB*, pages 1253–1256, 2004.

[16] Sara C. Madeira and Arlindo L. Oliveira. Biclustering algorithms for biological data analysis: A survey. *IEEE/ACM Trans. Comput. Biol. Bioinformatics*, 1(1):24–45, 2004.

[17] Francois Rioult and Arnaud Soulet. Average number of frequent (closed) patterns in bernouilli and markovian databases. In *ICDM'05*, pages 713–716, 2005.

[18] H. Toivonen. Sampling large databases for association rules. In *22nd VLDB Conf.*, 1996.

[19] Takeaki Uno, Masashi Kiyomi, and Hiroki Arimura. Lcm ver.3: collaboration of array, bitmap and prefix tree for frequent itemset mining. In *OSDM '05: Proceedings of the 1st international workshop on open source data mining*, pages 77–86, 2005.

[20] Craig Utley. Microsoft sql server 9.0 technical articles: Introduction to sql server 2005 data mining. http://technet.microsoft.com/en-us/library/ms345131.aspx.

[21] Guizhen Yang. The complexity of mining maximal frequent itemsets and maximal frequent patterns. In *KDD '04: Proceedings of the 2004 ACM SIGKDD international conference on Knowledge discovery and data mining*, 2004.

[22] Takeshi Yoshizawa, Iko Pramudiono, and Masaru Kitsuregawa. SQL based association rule mining using commercial RDBMS (IBM db2 UBD EEE). In *Data Warehousing and Knowledge Discovery*, pages 301–306, 2000.

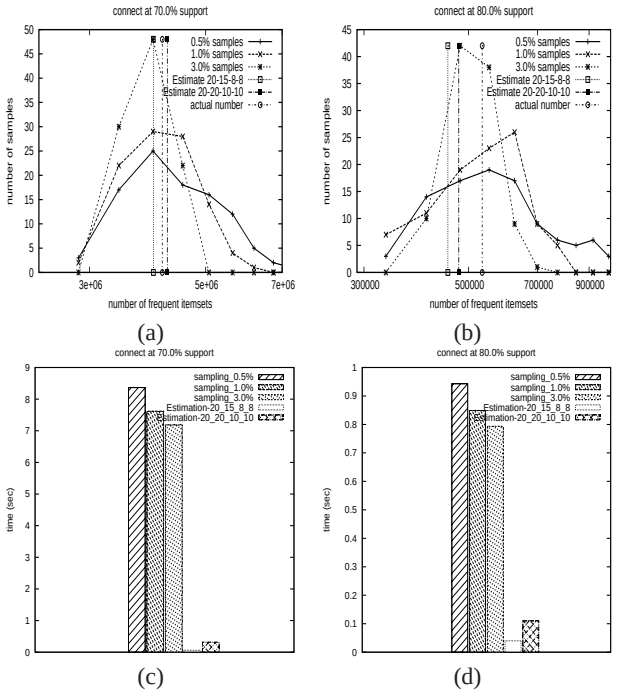


Figure 4: Sample Estimator vs. Sketch Matrix Estimator (Connect)

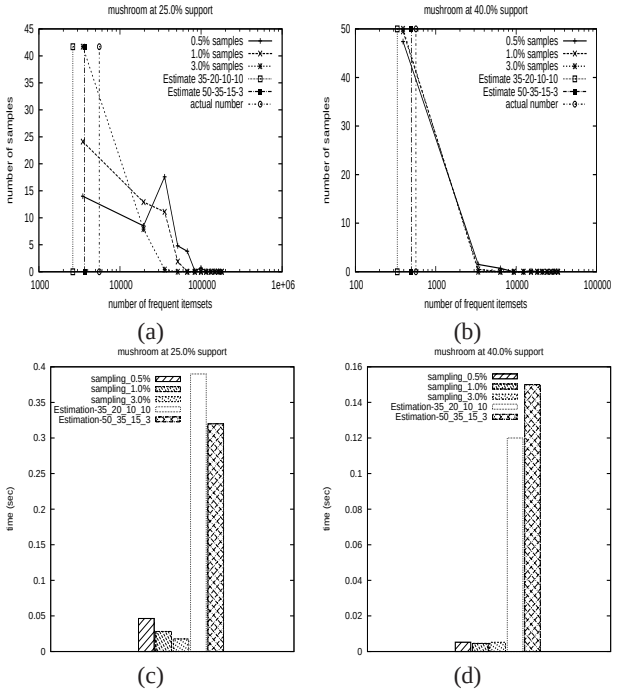


Figure 5: Sample Estimator vs. Sketch Matrix Estimator (Mushroom)

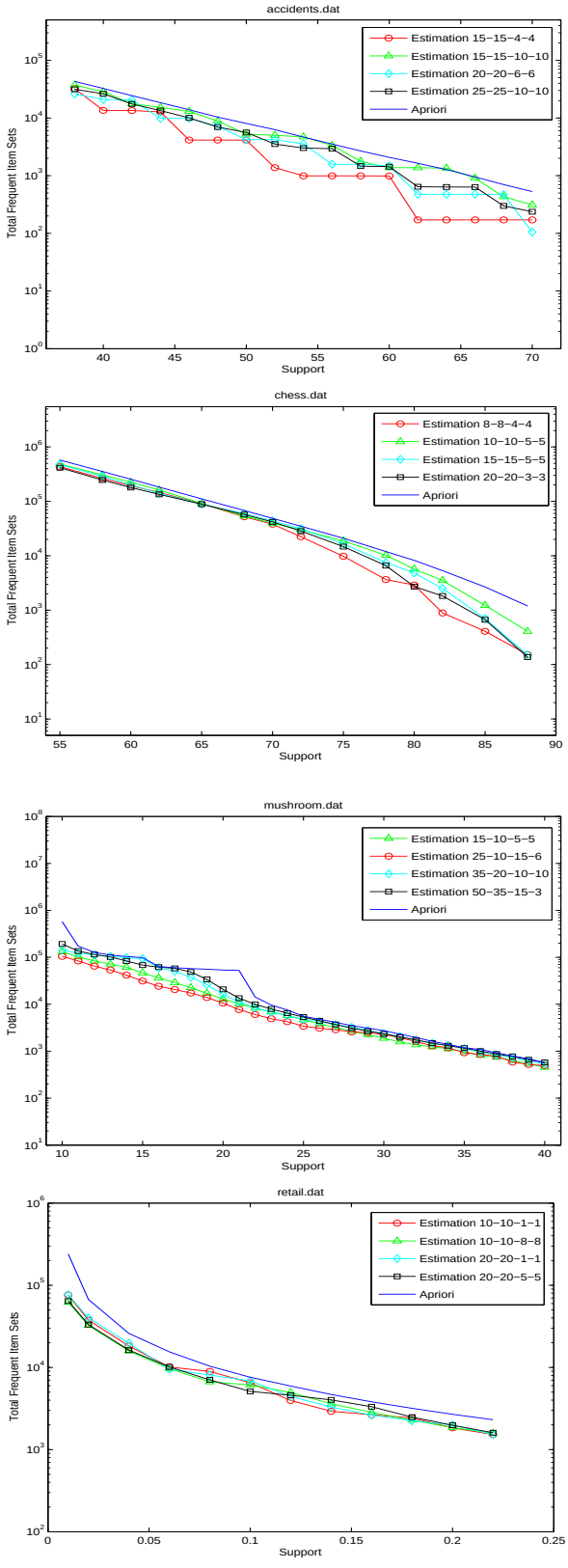


Figure 6: Estimation of Total Number of Frequent Itemsets