ON ADAPTIVE SAMPLING

Philippe Flajolet
I.N.R.I.A Rocquencourt
F-78150 Le Chesnay (France)

Abstract. We analyze the storage/accuracy trade–off of an adaptive sampling algorithm due to Wegman that makes it possible to evaluate probabilistically the number of distinct elements in a large file stored on disk.

1 Introduction

A problem that naturally arises in query optimization of data base systems [1] is to estimate the *number of distinct elements* (also called *cardinality*) of a large collection of data with unpredictable replications. The trivial solution that consists in building a list of distinct elements is usually too much resource consuming both in terms of storage and processing time requirements.

In [4] the authors have presented a solution called *Probabilistic Counting* that estimates the cardinality of a large file typically stored on disk; when using m words of in–core memory the algorithm presents an expected relative accuracy close to

$$\frac{0.78}{\sqrt{m}},$$

and it performs only a constant number of operations per element of the file. Wegman [11] has proposed an interesting alternative solution to that problem based on *Adaptive Sampling* techniques that is of comparable structural complexity. The Adaptive Sampling algorithm is also probabilistic in nature. We establish here that its expected relative accuracy is close to

$$\frac{1.20}{\sqrt{m}}$$

when m words of memory are used, so that its accuracy is roughly 50% less than that of Probabilistic Counting. The method however has some advantages in terms of processing time and of conceptual simplicity. It is also totally free of non–linearities when estimating the cardinalities of small files, a feature that may prove useful in several aplications. (In contrast, Probabilistic Counting in only *asymptotically* unbiased.)

Astrahan *et al.* [1] report on their experience with implementing Probabilistic Counting and Adaptive Sampling in the context of IBM's database system R. In terms of processing time, these probabilistic algorithms typically outperform standard sorting methods by a factor of about 8. In terms of storage consumptions, our formulæ show that using 100 words of memory will provide for a typical accuracy of 12% for Adaptive Sampling (8%, asymptotically, for Probabilistic Counting). This is to be contrasted again with sorting, where the auxiliary memory required has to be at least as large as the file itself! Some simulation results on Adaptive Sampling that support our analysis are presented in Section 4.

2 Wegman's Adaptive Sampling Method

The problem discussed here is the following. We are given a large collection F of data (typically a subset of a data base) which consists of records belonging to a given universe $\mathbf{U}$ (e.g. alphanumerical strings with length ≤ 20, if we consider some name fields). File F consists of data with "unpredictable" replications, in the sense that no statistical data model accounting for replications is available or applicable.

Sorting, eliminating duplicates then counting what remains is of course a solution, but it has the obvious disadvantages already mentioned in the introduction.

In contrast to sorting and like Probabilistic Counting, Wegman's Adaptive Sampling method —AS for short— is based on observing bits of hashed values of records scanned. We thus assume a hashing function is given that hashes elements of the universe of records $\mathbf{U}$ into sufficiently long bit streams. The algorithm is probabilistic in the sense that the result depends on the way the data (from the input file F) behaves with respect to the particular hash function selected. Accordingly, our analysis will also be probabilistic. (See Section 3 for a discussion of the analytic model.)

The algorithm also depends on the choice of a sequence of "properties" —i.e. *sets*— of bit streams $P_0, P_1, \ldots$, of which we shall soon fix a particular instance. Properties P_j are assumed to be such that $P_0 \supset P_1 \supset P_2 \cdots$ with

```
program AdaptiveSampling;
{Estimates the cardinality of a file F using a list of samples, LIST}.
const m = 64;
{accuracy is 1.20/√m; m = 64 gives about 15% accuracy}
var F: file of records; LIST, TEMPLIST : list of records;
    x : records; y : bitstream; depth : integer;
procedure hash(x : records) : bitstream; external;
begin
    depth := 0;
    for x in F do begin {Main scan loop}
        if (hash(x) ∈ 0^depth (0 + 1)*) then
            if not (hash(x) ∈ LIST) then
                LIST = LIST ∪ {hash(x)};
        if |LIST| > m then{Increase depth and split}
            repeat
                depth := depth + 1; TEMPLIST := ∅;
                for y in LIST do
                        if (y ∈ 0^depth (0 + 1)*) then
                            TEMPLIST := TEMPLIST ∪ {y};
                LIST := TEMPLIST;
            until |LIST| ≤ m; {Splitting done!}
    end; {Main scan loop}
    return(2^depth × |LIST|);
end.
```

Figure 1. Wegman's Adaptive Sampling Algorithm

the further condition,

$$\mathrm{Proba}\{x \in P_j\} = 2^{-j}.$$

(P_0 is the universal predicate.)

3 At every stage the algorithm keeps a *list* of at most m *sampled* (distinct) hashed values, where m is a design parameter that determines the accuracy of the method, and an integer index δ that corresponds to the current *depth of sampling*. Algorithm AS starts in phase 0 (depth is $\delta = 0$) by building a list without replications of hashed values of records encountered until $(m+1)$ such values have beeen found: At this time, the list of samples overflows; depth is increased to $\delta = 1$, the list is scanned and only those hashed values that satisfy P_1 are kept. Next, we resume scanning the file; the list is updated by appending those hashed values of elements that now

RECORD	HASHED	LIST of SAMPLES	depth	Estim#	Exact#
UDINE	10101	{10101}	0	1	1
NICE*	00101	{10101,00101}	0	2	2
PARIS	11011	{10101,00101,11011}	0	3	3
BORDE	01001	{00101,01001}	1	4	4
NAFPL	11101	{00101,01001}	1	4	5
PARIS	11011	{00101,01001}	1	4	5
BORDE	01001	{00101,01001}	1	4	5
MARSE	01010	{00101,01001,01010}	1	6	6
RENNE	10100	{00101,01001,01010}	1	6	7
LEIPZ	00010	{00101,00010}	2	8	8
CAEN*	10001	{00101,00010}	2	8	9
QUEBE	00111	{00101,00010,00111}	2	12	10
MARSE	01010	{00101,00010,00111}	2	12	10
CAEN*	10001	{00101,00010,00111}	2	12	10
PISA*	00100	{00010}	3	8	11

Figure 2. A typical execution of Adaptive Sampling with $m = 3$. File F consists here of 15 *records* (1st column) that are city code names in the form of alphanumerical strings; the corresponding *hashed* values over $\ell = 5$ bits appear in column 2. The third column shows the evolution of the *LIST* which keeps the elements that have been sampled. Column 4 displays sampling *depth* at each stage, with the running *estimates* for cardinality in column 5, and the *exact* cardinalities in the last column. The final estimate provided by Adaptive Sampling for the cardinality of F is 8 while the exact cardinality is 11.

satisfy P_1 and we continue until the list again overflows (reaches cardinality $m + 1$). At this point, the process repeats itself: We only retain those elements that satisfy property P_2, increase the depth to $\delta = 2$ etc.

In this way, at a phase where depth has value δ, the list keeps all elements (or rather their hashed values) that satisfy P_δ, so that if ℓ is the cardinality of the list, quite naturally, we may propose

$$2^\delta \cdot \ell$$

as an estimate of the cardinality of the file size.

One particularly simple way of implementing AS consists in choosing for P_j the set $\{0^j(0 + 1)^*\}$ consisting of all bit streams that begin with a sequence of at least j 0-bits. A specification of the corresponding algorithm is given in Figure 1 and an example of execution is displayed on Figure 2. We shall henceforth assume that this choice of P_j's has been made. (One could have used further randomization on the P_j by taking as P_j the set of binary streams that start with $b_1 b_2 \dots b_j$ for some randomly preselected b_1, b_2, etc.)

The idea underlying AS has relations to dynamic hashing of Larson [8] and extendible hashing of Fagin *et al.* [3], since it amounts to keeping only one page of the file under either of these algorithms.

3 Analysis

The analysis which we provide for Adaptive Sampling is made under the following probabilistic model: Hashed values of records are infinitely long bit streams that are independently and uniformly distributed over $\{0,1\}^\infty$. This is of course a simplification of reality; it is equivalent to assuming a uniform hash function from the universe $\mathbf{U}$ of records to infinite binary strings.

In other words, for analysis purpose, we assume that instead of elements of F, we are provided directly with a number n (unknown, to be estimated on the fly) of *random uniform infinitely long binary strings.*

The uniformity assumption is normally, with a careful implementation, justified since empirical studies (see [9] or [4] and references therein) confirm that this model matches reality extremely well under the following two condition: (i) a "reasonable" hashing function is used, like multiplicative hashing; (ii) enough randomness is available in the data in the sense that the hash function is used as an "information reduction" function (i.e. $2^\ell \ll |\mathbf{U}|$ where ℓ is the actual finite length of hashed values).

Considering hashed values to be infinitely long, i.e. assuming a collision free hash function, is harmless as long as we estimate cardinalities n such that $n \ll 2^\ell$ (say $100n \leq 2^\ell$), where ℓ is the actual finite length of hashed values.

Now that the probabilistic model of use has been specified,, we can introduce the analysis. The analysis will establish in passing that algorithm AS is *unbiased*, and more importantly that the relative accuracy is expected to be close to $1.20/\sqrt{m}$. The analysis decomposes into 4 phases.

A. *Recurrences.* The cardinality estimate provided by adaptive sampling can also be viewed as a recursively defined parameter of sets of binary strings, whence recurrences for expected values.

B. Generating Functions. The recurrences for the mean value of the estimate of AS and its standard deviation (actually second moment) are solved by introducing suitable generating functions.

C. Elementary Approximations. These reduce the standard deviation of the estimate to a simpler asymptotic form.

D. Mellin Transforms. As is customary with analyses of these sort, there are some hidden periodicities which, although not numerically important, render the analytic process more intricate. The Mellin integral transform turns out to be the method of choice for the final asymptotic estimates.

We now execute this programme.

A. Recurrences. By construction, algorithm AS is insensitive to the structure of replications in the file operated upon. If n hashed values (bit streams) are drawn according to the previously defined model, then the probability that k of these start with a 0-bit is the *Bernoulli probability*:

$$B_{n,k} = \frac{1}{2^n}\binom{n}{k}. \tag{1}$$

Let ω be a finite subset of $\{0,1\}^\infty$; denote by $\omega/0$ the set:

$$\omega/0 = \{y \in \{0,1\}^\infty \mid 0y \in \omega\}$$

with a similar definition for $\omega/1$. Let $K(\omega)$ be the estimate provided by algorithm AS. Then K admits the inductive definition:

$$K(\omega) = \begin{cases} 2K(\omega/0) & \text{if } |\omega| > m \\ |\omega| & \text{otherwise.} \end{cases} \tag{2}$$

Let K_n denote the expectation of the random variable $K(\omega)$ when a random n-subset ω of $\{0,1\}^\infty$ is chosen; similarly let L_n denote the expectation of $K^2(\omega)$, that is the second moment of K. From the recursive definition (2) with the expression (1) for the Bernoulli probabilities, we find the following relations valid for $n > m$,

$$K_n = 2\sum_{k\geq 0} \frac{1}{2^n}\binom{n}{k} K_k \ , \tag{3}$$

$$L_n = 4 \sum_{k \geq 0} \frac{1}{2^n} \binom{n}{k} L_k \ , \tag{4}$$

together with the initial conditions: $K_n = n$, $L_n = n^2$, when $n \leq m$. These permit already to determine numerically the exact values of K_n and L_n.

B. Generating Functions. We proceed by introducing the corresponding *exponential generating functions*:

$$K(x) = \sum_{n \geq 0} K_n \frac{x^n}{n!} \,; \quad L(x) = \sum_{n \geq 0} L_n \frac{x^n}{n!}.$$

In this way, Equations (3), (4) translate into the difference equations:

$$K(x) = 2e^{x/2} K(x/2) + a(x), \tag{5}$$

$$L(x) = 4e^{x/2} L(x/2) + b(x), \tag{6}$$

for two polynomials $a(x)$ and $b(x)$ of degree at most m that are easily determined from the initial conditions. We find

$$a(x) \equiv 0 \,; \quad b(x) = e_{m-1}(x),$$

where $e_m(x)$ denotes the truncated exponential:

$$e_m(x) = \sum_{j=0}^{m} \frac{x^j}{j!} \, .$$

The solution to equation (5) with the corresponding initial conditions is easily checked to be:

$$K(x) = xe^x,$$

so that we have, as anticipated, $K_n = n$.

Proposition 1 *Adaptive Sampling (AS) is unbiased in the sense that when the input are uniformly distributed binary streams, the expectation of the estimate of the cardinality of a file that it provides is equal to the cardinality of the file.*

We now turn to the more interesting problem of estimating the accuracy of algorithm *AS*. To that purpose, in order to be able to solve generating function equations by iteration, we introduce the modified function:

$$L_1(x) = L(x) - xe^x - x^2 e^x \, .$$

That function satisfies the equation:

$$L_1(x) = 4e^{x/2}L_1(x) + x(e^x - e_{m-1}(x)) \ . \tag{7}$$

Since we have $L_1(x) = O(x^3)$ as $x \to 0$, equation (7) can now be solved by iteration and we find (see [6] for a general framework):

$$\begin{aligned} L_1(x) &= \sum_{k\geq 0} 4^k e^{x(1-1/2^k)} \frac{x}{2^k}[e^{x/2^k} - e_{m-1}(x/2^k)] \\ &= x \sum_{k\geq 0} 2^k [e^x - e^{x(1-1/2^k)} e_{m-1}(x/2^k)] \ . \end{aligned} \tag{8}$$

From (8), we can compute Taylor coefficients explicitly and we get[1]:

$$L_{1,n} \equiv n!\,[x^n]L_1(x) = n \sum_{k\geq 0} 2^k [1 - \beta_{n-1,m-1}(1/2^k)],$$

where $\beta_{n,m}$ denotes the *truncated binomial series*

$$\beta_{n,m}(a) = \sum_{j=0}^{m} \binom{n}{j} a^j (1-a)^{n-j} \tag{9}$$

corresponding to the initial terms of the binomial expansion of $((1-a)+a)^n$.

Since $L_n = L_{1,n} + n^2$ and since $K_n = n$, we have:

Proposition 2 *The variance of the estimate of algorithm AS when applied to n uniformly drawn binary streams is given by*

$$V_n = n \sum_{k\geq 0} 2^k [1 - \beta_{n-1,m-1}(1/2^k)] \ . \tag{10}$$

C. Elementary Approximations. Similar sums appear not too unexpectedly in the analysis of Dynamic and Extendible Hashing Algorithms [10]. The standard route starts by replacing V_n in the above formula by an *exponential approximation*

$$(1-\alpha)^n \approx e^{-n\alpha},$$

[1] We let as usual $[x^n]\,f(x)$ denote the coefficient of x^n in the Taylor expansion of $f(x)$ at 0.

and using this inside formula (10) can be justified easily following the lines of [7, p.131]. This gives rise to the estimate

$$V_n = v_n + o(n^2),$$

with

$$v_n = n \sum_{k \geq 0} 2^k [1 - e^{-n/2^k} e_{m-1}(n/2^k)] \ .$$

Notice that v_n represents the variance of the estimate when the number of input streams is a random variable that obeys a Poisson Law with parameter n.

D. Mellin Transforms. At this point, we plunge into analysis, using Mellin transform techniques, a route again inspired by the corresponding treatment in [7]. (See also [5] for a more general presentation of the method.)

The Mellin transform of a real function $f(x)$ is a complex function denoted as $f^*(s)$ and defined by

$$F^*(s) = \int_0^\infty F(x) x^{s-1} dx.$$

We consider the real function:

$$F(x) = \sum 2^k [1 - e^{-x/2^k} e_{m-1}(x/2^k)] \ .$$

Its Mellin transform $F^*(s)$ exists for $-2 < \Re(s) < -1$ and is easily determined using basic principles [2], as we now explain.

1. The Mellin transform of e^{-x} is the *Gamma* function and more generally, one has

   $$\int_0^\infty [1 - e^{-x} e_{m-1}(x)] x^{s-1} dx \ = \ \Gamma(s) \binom{s+m-1}{m-1},$$

   an equation valid for $-m < \Re(s) < 0$.

2. The transform of $f(ax)$ is $a^{-s} f^*(s)$, so that formally the transform of a sum $\sum_k \alpha_k f(\gamma_k x)$ is $(\sum_k \alpha_k \beta_k^{-s}) \cdot f^*(s)$.

From these observations, we get the Mellin transform of $F(x)$, namely

$$F^*(s) = -\Gamma(s) \binom{s+m-1}{m-1} \frac{1}{1 - 2^{1+s}} \,. \tag{11}$$

Using the familiar inversion theorem for Mellin transforms, we can recover $F(x)$ as

$$F(x) = \frac{1}{2i\pi}\int_{-3/2-i\infty}^{-3/2+i\infty} F^*(s)x^{-s}ds, \tag{12}$$

and classically obtain terms of the asymptotic expansion of $F(x)$ by moving the line of integration to the right (say to the line $\Re(s) = 10$) only taking residues into account. In this way, denoting by $\mathrm{Res}[h(s)]$ the residue of function $h(\cdot)$ at s, we get:

$$F(x) = -\sum_s \mathrm{Res}[F^*(s)x^{-s}] + O(x^{-10}).$$

There the sum is extended to all poles of F^* in the strip $-3/2 < \Re(s) < 10$, that is to the points:

$$\sigma_0 = -1\ ; \quad \sigma_k = -1 + 2ik\pi/\log 2, \quad k \in \mathbf{Z} \setminus \{0\}\ .$$

At $s = \sigma_0 \equiv -1$, the residue of $F^*(s)$ is found to be equal to

$$-1/((m-1)\log 2).$$

Computing other residues in a similar fashion we get the asymptotic expansion of $F(x)$ towards infinity whence the corresponding result for v_n and finally V_n. We find:

Theorem 1 *The variance of Adaptive Sampling when applied to n random binary streams satisfies the relation*

$$\frac{V_n}{n^2} = \frac{1}{(m-1)\log 2} + P(\log_2 n) + o(1),$$

where $P(u)$ is a periodic function of u with mean value 0 *and Fourier expansion $P(u) = \sum_{k\in\mathbf{Z}\setminus\{0\}} p_k e^{-2iku}$ such that*

$$p_k = \frac{1}{\log 2}\Gamma(-1 + \frac{2ik\pi}{\log 2})\binom{2ik\pi/\log 2 + m - 2}{m-1}.$$

4 Conclusions

If we neglect the periodic fluctuations in the variance of the estimate provided by Adaptive Sampling (the amplitudes of these fluctuations are as

m	*AS-Text*	*AS-Rand*	$\Pi(m)$	*PC*
8	37.6%	42.4%	42.4%	31.9%
16	25.3%	30.2%	30.0%	19.3%
32	17.8%	21.6%	21.2%	12.9%
64	13.8%	14.6%	15.0%	9.6%
128	10.5%	10.8%	10.6%	6.6%
256	6.7%	7.3%	7.5%	4.65%

Figure 3. A comparison of the empirical standard error of Adaptive Sampling on textual data (*AS-Text*) or random numerical data (*AS-Rand*), against the theoretical prediction $\Pi(m)$ given in Eq. (13), and against corresponding simulations for Probabilistic Counting (*PC*).

---

usual very small; alternatively, we "average" the values of the variance considering that $\log_2 n$ is uniformly distributed modulo 1), we find the approximate expression $V_n \approx n^2/((m-1)\log 2)$. What is of interest in the context of probabilistic estimation algorithms of this sort is the "standard error" (a measure of the expected relative error) defined as the quotient of the standard deviation of the estimate by the exact value n, i.e. $V_n^{1/2}/n$. This quantity is a function of m, with little dependence on n, as is seen from the asymptotic form of V_n given by Theorem 1. As expected, if we use more memory (i.e. m gets larger), the accuracy of the results is going to be better. Summarizing our previous discussion, we have established:

Fact 1 *The accuracy[2] of Adaptive Sampling measured by the standard error, when m words of memory[3] are used is closely approximated by the formula*

$$\Pi(m) \approx 1.20/\sqrt{m} \tag{13}$$

We have conducted several experiments on actual text files (*AS-Text* in Fig. 3) representing on-line documentation available on one of our systems. The files range in size from a few kilobytes to about half a megabyte with cardinalities (there records are lines of text) in the range 1000–17000. To

[2] Observe that the accuracy is defined here in terms of a standard deviation, that is using a quadratic (L^2) norm. When compared to the expected error in the sense of the L^1 norm, our definition provides a slightly pessimistic estimate.

[3] With machine word lengths ≥ 32 and file sizes $\leq 10^9$, we may freely asume that a hashed value fits in a single word.

each of the 8 files, 9 different multiplicative hashing functions have been applied resulting in a total of 72 simulations for each value of m. We have considered the following values of m: 8, 16, 32, 64, 128, 256. In addition, for each of these values of m, we have conducted 600 simulations on files obtained from random number generators (*AS-Rand* in Fig. 3) with 100 simulations for cardinalities equal to 5000, 6000, 7000, 8000, 9000, 10000.

These simulations validate our estimates: no detectible bias occurs, and the observed relative errors are very close to the predicted value $\Pi(m)$ given by formula (13).

For reference, results are also compared with those of Probabilistic Counting (PC) as given in [4], themselves based on 160 simulations (16 files $\times$ 10 hashing functions).

Conclusions. Adaptive Sampling is an unbiased estimator of cardinalities of large files that necessitates minimal auxiliary storage and processes data in a single pass. Theoretical predictions on the accuracy of the method based on the formula $1.20/\sqrt{m}$ match reality quite well. Adaptive Sampling appears to be about 50% less accurate than Probabilistic Counting when using comparable memory size (the m parameter); accordingly, to attain the same accuracy, AS would need to use about twice as much space. However algorithm AS is completely free of non-linearities for smaller values of cardinalities n; it may also have some advantage in terms of processing time on large files, since then the computations in the inner loop hardly ever require more than hashing and a simple test, while Probabilistic Counting requires also one address computation and an update of a $BITMAP$ vector.

Acknowledgement. The author would like to express his gratitude to Mark Wegman for sharing his ideas concerning Adaptive Sampling and for stimulating discussions that led to the present work.

References

[1] M. M. Astrahan, M. Schkolnick, and K-Y. Whang. Approximating the number of unique values of an attribute without sorting. *Information Sciences*, 12:11–15, 1987.

[2] G. Doetsch. *Handbuch der Laplace Transformation,* Vol. 1–3. Birkhäuser Verlag, Basel, 1955.

[3] R. Fagin, J. Nievergelt, N. Pippenger, and R. Strong. Extendible hashing: A fast access method for dynamic files. *A.C.M. Trans. Database Syst.*, 4:315–344, 1979.

[4] P. Flajolet and G. N. Martin. Probabilistic counting algorithms for data base applications. *J. of Computer and System Sciences*, 31:182–209, 1985.

[5] P. Flajolet, M. Régnier, and R. Sedgewick. Some uses of the Mellin integral transform in the analysis of algorithms. In A. Apostolico and Z. Galil, editors, *Proceedings of NATO Advanced Study Institute on Combinatorial Algorithms on Words*, pages 241–254. Springer Verlag (NATO ASI Series, Vol. F12), 1985.

[6] P. Flajolet, M. Régnier, and D. Sotteau. Algebraic methods for trie statistics. *Annals of Discr. Math.*, 25:145–188, 1985.

[7] D. E. Knuth. *The Art of Computer Programming*, volume 3: Sorting and Searching. Addison-Wesley, 1973.

[8] P. A. Larson. Dynamic hashing. *BIT*, 18:184–201, 1978.

[9] V. Y. Lum, P. S. T. Yuen, and M. Dodd. Key to address transformations: A fundamental study based on large existing format files. *Comm. ACM*, 14:228–239, 1971.

[10] M. Régnier. Evaluation des performances du hachage dynamique, 1983. Thèse de 3e cycle, Université Paris–Sud.

[11] M. Wegman. Sample counting, 1984. Private Communication.