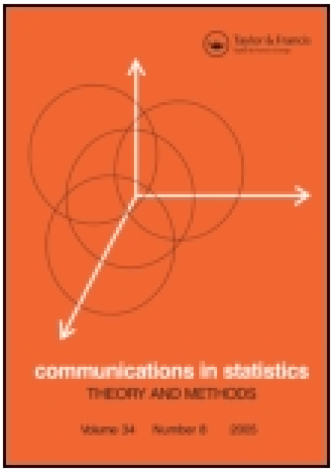

Communications in Statistics - Theory and Methods
Publication details, including instructions for authors and subscription information:
http://www.tandfonline.com/loi/lsta20

Computing the probability of hash table/urn overflow

M.V. Ramakrishna [a]
[a] Computer Science Department , Michigan State University , East Lansing, 48824-1027, Michigan
Published online: 27 Jun 2007.

COMMUN. STATIST.-THEORY METH., 16(11), 3343-3353 (1987)

COMPUTING THE PROBABILITY OF HASH TABLE / URN OVERFLOW

M.V. Ramakrishna

Computer Science Department
Michigan State University
East Lansing, Michigan 48824-1027

Key Words and Phrases: balls and urns model; combinatorial extreme- value distributions; classical occupancy problem; perfect hashing.

ABSTRACT

We analyze the probability of a random distribution of n balls into m urns of size b resulting in no overflows. This solves the computational problem associated with a classical combinatorial extreme-value distribution. The problem arose during the analysis of a technique, called perfect hashing, for organizing data in computer files. The results and techniques presented can be used to solve several problems in the analysis of hashing techniques.

1. INTRODUCTION

Consider a traditional urn model. There are n balls to be randomly distributed into m urns, each urn having a capacity of at most b balls. Let each ball be randomly tossed into an urn so that the probability of a ball falling into a particular urn is $1/m$ and independent of the outcome of other tossings. If an urn already contains b balls, any subsequent ball tossed into the urn is said to overflow. Let $P(n,m,b)$ denote the probability of a random distribution of n balls into m urns of size b resulting in no overflows.

One of the combinatorial extreme-value problems considered by David and Barton and Barton and David is as follows[1, 3]. When n balls are randomly distributed among m urns, let X denote the number of balls in the urn(or urns) containing the maximum number of balls. Then the cumulative probability distribution of X, $\mathrm{pr}\{X \le b\}$ is precisely $P(n,m,b)$ defined above. Barton and David consider several similar combinatorial extreme-value problems. One common feature of these problems is that "the probability distribution functions are difficult to evaluate even for moderate sized samples"[1, p 63]. However, "there are a number of problems in which its ($P(n,m,b)$) enumeration is of interest"[3, p 221]. In this paper we solve the classical computational problem by giving a simple recurrence relation which enables easy and exact computation of $P(n,m,b)$. We also study how the values of $P(n,m,b)$ computed using a simple approximation approach the exact values of $P(n,m,b)$.

Kolchin, Sevast'yanov and Chistyakov have given some results about the asymptotic behavior of $P(n,m,b)$ [5, pp 96-115]. However, to the best of our knowledge the computational problem we have addressed in this paper has not been solved so far. We encountered this classical problem while analyzing *perfect hashing*. Hashing refers to a class of techniques for organizing files stored in computer memory. Exact and efficient computation of $P(n,m,b)$ was essential for the analysis. The results and techniques presented in this paper also enable us to answer open problems in the analysis of some other hashing schemes (see sections 6 and 7).

2. BACKGROUND

Let $F(n,m,b)$ denote the number of ways in which n balls can be distributed among m urns so that no urn receives more than b balls (assume $n \le mb$). It follows that

$$P(n,m,b) = \frac{F(n,m,b)}{m^n}. \tag{1}$$

For $b = 1$ the expression for $F(n,m,b)$ is trivial. When $b > 1$ the analysis is difficult. David and Barton and Barton and David give the following expression for $F(n,m,b)$ [1, 3]:

$$F(n,m,b) = \sum_{0 \le f_i \le b} \left[n! \Big/ \prod_{i=1}^{m} f_i! \right], \qquad n \le mb,$$

where f_i denotes the number of balls in the i th urn and the summation is over all possible combinations of f_i such that $\sum_{i=1}^{m} f_i = n$. An example makes the above expression clear. $F(4,3,2)$ denotes the number of ways in which 4 balls can be distributed among 3 urns, each capable of holding at most 2 balls:

$$F(4,3,2) = \frac{4!}{2!1!1!} + \frac{4!}{2!2!0!} + \frac{4!}{0!2!2!} + \frac{4!}{1!2!1!} + \frac{4!}{2!0!2!} + \frac{4!}{1!1!2!} = 54$$

David and Barton (1962) point out "$F(N)$ [$= F(n,m,b)$] does not posses a simple form"[3, p 221]. The formula is not suited for numerical evaluation of $F(n,m,b)$. There is a generating function used to compute $F(n,m,b)$:

$$F(n,m,b) = \text{Coefficient of } (x^n/n!) \text{ in } \left[G_b(x)\right]^m, \quad \text{where}$$

$$G_b(x) = (1 + x/1! + x^2/2! + \cdots + x^b/b!).$$

Computations using the above generating function involve the handling of very large integers (of the order of $b!^m$). This is neither suitable for hand calculations nor for computer evaluation of $P(n,m,b)$, even for moderate values of parameters. Evaluation of $F(n,30,30)$ took several hours of computer time on the MAPLE symbolic algebra system(capable of handling very large integers) running on a VAX-780 processor[2]. For larger values of m and b, it is prohibitively expensive to compute $F(n,m,b)$ using the generating function. It appears that the computational complexity can be reduced to some extent using Fast Fourier Transforms (In this regard, there is a note by Monahan appearing at the end of this paper). In the following sections we present an efficient and simple solution to the problem. A procedure is given which enables computation of a table of $P(n,m,b)$ values, having approximately $mn/2$ entries, using only six arithmetic operations per each value.

3. RECURRENCE RELATION FOR $P(n,m,b)$

Suppose that n balls have already been randomly distributed among m urns and no overflow has occurred. Let the next ball, the $(n+1)$st, be tossed into a randomly chosen urn. We use $R(n+1,m,b)$ to denote the conditional probability that the $(n+1)$st ball will not overflow. Then $R(n+1,m,b)$ can be expressed as

$$R(n+1,m,b) = \frac{P(n+1,m,b)}{P(n,m,b)}. \tag{2}$$

The $(n+1)$st ball will overflow if and only if it falls into an urn already full (i.e., one containing b balls). The probability of this event is the same as the probability of an arbitrary but fixed urn being full. Hence, the probability of the $(n+1)$st ball overflowing can be expressed as

$$1-R(n+1,m,b) = \frac{\binom{n}{b} F(n-b,m-1,b)}{F(n,m,b)}. \tag{3}$$

The numerator represents the total number of combinations resulting in the fixed urn being full. The term $\binom{n}{b}$ represents the number of ways in which b balls (those in the full urn) can be chosen from n balls. The number of ways in which the remaining $(n-b)$ balls can be distributed among the other $(m-1)$ urns is given by $F(n-b,m-1,b)$. By combining equations (1), (2) and (3) we obtain

$$P(n+1,m,b) = P(n,m,b) - \binom{n}{b} P(n-b,m-1,b) \frac{(m-1)^{n-b}}{m^n} \tag{4}$$

To evaluate $P(n,m,b)$ using (4) we need to calculate $P(i,j,b)$ for $j = 1,2,\cdots,m$, and $i = 1,2,\cdots,n-(m-j)b+1$. It should be noted that similar computations are required implicitly by the generating function approach. Although the above recurrence relation looks complicated, involving large numbers of the order of m^n, the computation can be organized so that the evaluation of each new value of $P(i,j,b)$ requires only 6 arithmetic operations. Appendix A contains a

TABLE I. Cumulative probability distribution, $P(400,30,b)$

b	15	16	17	18	19	20	21
$P(n,m,b)$	0.0000	0.0001	0.0042	0.0441	0.1717	0.3767	0.5895
b	22	23	24	25	26	27	28
$P(n,m,b)$	0.7571	0.8675	0.9321	0.9670	0.9846	0.9931	0.9970
b	29	30	31	32	33		
$P(n,m,b)$	0.9988	0.9995	0.9998	0.9999	1.0000		

procedure (suitable for hand calculation or computer evaluation) to compute a table of $P(n,m,b)$ for a fixed b and $1 < m \le m_{max}$, $1 \le n \le bm_{max}$. The procedure is based on (4) and the following identity:

$$\binom{n}{b}\frac{(m-1)^{n-b}}{m^n} = \left(\frac{n}{n-b}\right)\left(\frac{m-1}{m}\right)\left\{\binom{n-1}{b}\frac{(m-1)^{n-b-1}}{m^{n-1}}\right\} \tag{5}$$

The iterations start with the initializations $P(n,m,b) = 1.0$, for $1 \le n \le b$, $1 \le m \le m_{max}$. Denote the value of $\binom{n}{b}\frac{(m-1)^{n-b}}{m^n}$ by *term*. Initially when n is equal to b, the value of *term* is $1/m^b$. Each subsequent value of *term* can be obtained by multiplying the previous value of *term* by $(\frac{m-1}{m})(\frac{n}{n-b})$ and hence a total of 6 arithmetic operations are sufficient to compute the next value of $P(n,m,b)$. Thus for a given value of m and b the evaluation of all values of $P(i,j,b)$, $j = 1,2,\cdots,m$, and $i = 1,2,\cdots,n-(m-j)b+1$ requires a total number of arithmetic operations proportional to nm, and hence the procedure is optimal. For example, computation of $P(n,30,30)$ requires only a few seconds of computer time. Appendix A also contains a brief discussion of the numerical stability of the computation. Using the procedure, $P(n,m,b)$ can be computed even for large values of the parameters. For example, computation of $P(5000,500,20) = 0.4520$ poses no problem and requires approximately 70 seconds of VAX-780 computer time.

Table I shows the cumulative probability distribution of X, $\text{pr}\{X \le b\}$. The random variable X denotes the number of balls in the urn(urns) containing the largest number of balls when 400 balls are randomly distributed into 30 urns.

4. RECURRENCE RELATION FOR $R(n,m,b)$

Consider the computation of $R(n,m,b)$, the conditional probability of the nth ball not overflowing given that $n-1$ balls have been distributed into m urns of size b and none have overflowed (such computations are required in some applications [10]). It is straightforward to compute $R(n,m,b)$ which is equal to the ratio $P(n,m,b)/P(n-1,m,b)$, when the values of $P(n,m,b)$ is much larger than the machine-epsilon (see the note at the end of Appendix A for an explanation of machine-epsilon). Roundoff errors in the value of $R(n,m,b)$ so computed become severe as n approaches mb, when the corresponding $P(n,m,b)$ values approach the machine-epsilon. It is interesting to note that the lowest nonzero value of $R(n,m,b)$ is $1/m$, large compared to the machine-epsilon. (When $mb-1$ balls have been distributed and none have overflowed, precisely $m-1$ urns must be full and the other urn must contain $b-1$ balls. It then follows that $R(mb,m,b)=1/m$. When $n > mb$, $R(n,m,b)=0$ and $R(n,m,b)=1$ for $n \le b$). On the other hand, the corresponding $P(mb,m,b)=\frac{(mb)!}{(b!)^{mb}}$ is extremely small, of the order of $(m/b^{b-1})^{mb}$. (For the present assume that the range of m, b we are interested in is 5 to 100.) Thus, although the error in $P(n,m,b)$ may not be significant when its value is very small(typically of the order of 10^{-10}), the error in the value of $R(n,m,b)$ computed using (4),(5) and (2) is extremely large (even the values computed using double precision arithmetic are completely meaningless). This computational problem can be overcome by using a *reverse* recurrence relation for $R(n,m,b)$. Since we know the value of $R(n,m,b)$ when $n = mb$, the idea is that we should be able to overcome the computational difficulty by proceeding backwards starting from $n = mb$.

Replacing n by $n-1$ in (4) and dividing throughout by $P(n-1,m,b)$ we obtain

$$R(n,m,b) = 1 - \binom{n-1}{b} \frac{P(n-b-1,m-1,b)}{P(n-1,m,b)} \frac{(m-1)^{n-b-1}}{m^{n-1}}$$

Using (2), the identity (5) and eliminating $P(n,m,b)$ from the above equation we obtain the following recurrence relation.

$$\frac{1}{R(n-1,m,b)} = 1 + \frac{1-R(n,m,b)}{R(n-b-1,m-1,b)} \left(\frac{n-b-1}{n-1}\right)\left(\frac{m}{m-1}\right) \qquad (6)$$

Starting from $n = mb$, $R(mb,m,b)=1/m$, $R(n,m,b)$ for $n = mb-1, mb-2, \cdots, b$ can be computed using (6). The numerical stability problems mentioned before are not encountered when using this recurrence relation for computing $R(n,m,b)$. Since $P(n,m,b)=\prod_{i=1}^{n} R(i,m,b)$, it is a good heuristic to compute $P(n,m,b)$ and $R(n,m,b)$ using (4) for small values of n, and using (6) when n approaches mb.

5. APPROXIMATE FORMULA

In this section we obtain an approximate, closed form expression for $P(n,m,b)$. The main approximation is to assume that urns overflow independently when balls are tossed randomly into the urns.

Let $Pov(\alpha,b)$ denote the probability of an arbitrary but fixed urn overflowing when $n = \alpha mb$, $0 \le \alpha \le 1$, balls are randomly tossed into m urns, each having a capacity of b balls.

Using the Poisson approximation of the binomial distribution we can express Pov as the following sum:

$$Pov(\alpha,b) \approx \sum_{i=b+1}^{n} \frac{e^{-b\alpha}(b\alpha)^i}{i\,!}$$

where $b\alpha = n/m$ is the average number of balls per urn (This is a good approximation for moderate values of $b\alpha$) [4]. For large values of b, and α not too close to 1.0, the summation can be approximated as follows:

$$Pov(\alpha,b) \approx \frac{(b\alpha)^{b+1}}{(b+1)!} e^{-b\alpha}\left\{1 + \frac{b\alpha}{b+2} + \frac{(b\alpha)^2}{(b+2)(b+3)} + \cdots\right\}$$

$$\approx \frac{(b\alpha)^{b+1}}{(b+1)!} e^{-b\alpha}\left\{1 + \frac{b\alpha}{b+2} + \frac{(b\alpha)^2}{(b+2)^2} + \cdots\right\}$$

$$Pov(\alpha,b) \approx \frac{(b\alpha)^{b+1}}{(b+1)!} e^{-b\alpha}\left\{\frac{b+2}{b(1-\alpha)+2}\right\} \tag{7}$$

Under the assumption that urns overflow independently of each other, $P(n,m,b)$ expressed as a function of α, m and b is given by

$$P(n,m,b) \approx (1 - Pov(\alpha,b))^m \approx e^{-m\,Pov(\alpha,b)}. \tag{8}$$

If α is small compared to 1, the term $\frac{b+2}{b(1-\alpha)+2}$ in (7) above evaluates to approximately 1. The corresponding $P(n,m,b)$ given by (8) is precisely same as the result obtained by David and Barton [3, pp 238-239] [1, pp 73-74].

Figure I is a plot of the exact and the approximate value of $P(n,m,b)$. The parameter b is 5 for all the curves and there is one pair of curves for each m of 5, 10, 20, 40 and 80. We observe that (8) is a good approximation for $P(n,m,b)$ when m is above 80 (for the case of $b = 5$). For larger values of b, (8) begins to be a good approximation at a lower value of m.

The approximation for $P(n,m,b)$ given by (7) and (8) also helps us understand the effect of increasing m on the value of α to keep $P(\alpha,m,b)$ a constant at a given value. In figure I, every small percentage drop in α allows a doubling of m to keep $P(\alpha,m,b)$ constant at 0.2, say. Consider equation (7) and take its derivative:

$$\frac{d}{d\alpha} Pov(\alpha,b) \approx Pov(\alpha,b)\left\{\frac{b+1}{\alpha} - b\left(\frac{b+1-b\alpha}{b+2-b\alpha}\right)\right\}$$

$$\approx Pov(\alpha,b)\left[b\,(1/\alpha - 1)\right] \qquad \text{(for large } b \text{ and } \alpha \text{ not too near 1.0)}$$

This implies that a small change in α results in magnified, by a factor of $b(1/\alpha-1)$, change in the value of $Pov(\alpha,b)$. It follows from (8) that a small drop in α is sufficient to compensate a large increase in m to maintain the value of $P(\alpha,m,b)$ constant at a required value.

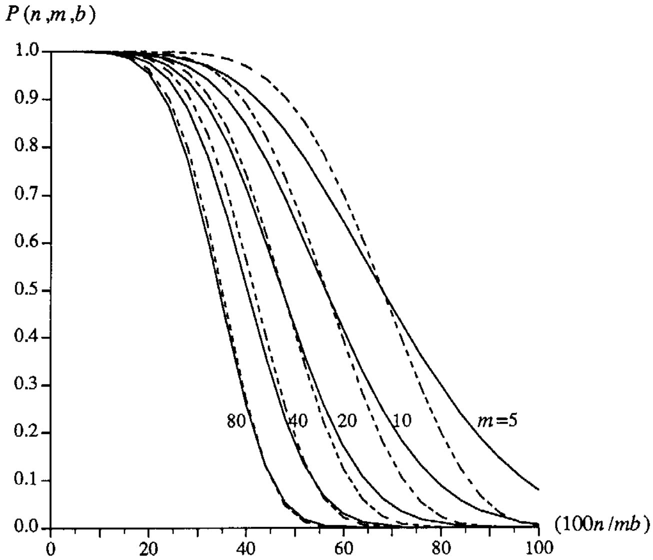

FIG. I. Comparison of exact and approximate values of $P(n,m,b)$.
Urn capacity (b) : 5, Number of urns (m) : 5,10,20,40,80
Solid lines : $P(n,m,b)$ computed using (8)
Dashed lines : exact values of $P(n,m,b)$

6. APPLICATION TO PERFECT HASHING

Consider a set of n integers $I = \{x_1, x_2, \cdots, x_n\}, I \subset \{1,2,\cdots,M\}$. Typically M is very large, of the order of 10^{10}. Each integer is referred to as a *key*. The problem is to store the n keys in m pages of computer memory, each page having a capacity to hold up to b keys, $n \leq mb$. In practice each key has some additional information associated with it. The key and its associated information is called a *record* and the set of records stored is called a *file*. The keys(records) in the file are to be organized in such a way that any given key x (and hence the associated record) can be efficiently retrieved. A hashing function h, $h: I \rightarrow [1,m]$, assigns each key an address in the range $1, \cdots, m$. Given a key x_i we compute $h(x_i)$ and store the record on that page. The record is said to hash into page $h(x_i)$. If more than b keys hash into a page, the page overflows. The overflowed keys from a page have to be stored elsewhere. One of the main issues in hashing is how to efficiently handle overflows.

A hashing function h is said to be a *perfect* hashing function if it causes no overflows (no more than b keys hash into every page). In [9] we consider the following trial-and-error method of finding perfect hashing functions for a given set of n keys to be hashed into m pages each of size b. Choose a function at random from the set of all functions mapping n objects to m objects.

Hash all the keys using the chosen function. If none of the pages receive more than b keys then we have found a perfect hashing function. Otherwise, choose another function at random and repeat the process until a function which is perfect for the given set is found. We define a trial as the process of choosing a function at random from the set of all functions and hashing the keys using the chosen function to verify if it is perfect. The probability of a trial succeeding is precisely $P(n,m,b)$ discussed in this paper.

$P(n,m,b)$ is a measure of the performance of the trial-and-error method of finding perfect hashing functions. The reciprocal of $P(n,m,b)$ gives the expected number of trials required to find a perfect hashing function. A trial may stop as unsuccessful immediately after one of the pages overflows. We define the expected cost of a trial as the number of hash function evaluations required to determine if the trial is successful.

Expected Cost of a Trial

Consider the balls and urns model. Let $E(n,m,b)$ denote the expected number of balls to be tossed before the first ball overflows. $E(n,m,b)$ is the same as the expected cost of a trial defined above. $E(n,m,b)$ is given by

$$E(n,m,b) = \sum_{i=1}^{n} i * \left\{ \begin{array}{l} \text{Probability that the } i \text{ th ball} \\ \text{overflows and none of the} \\ \text{balls } 1, \cdots, (i-1) \text{ overflow} \end{array} \right\} + n * \left\{ \begin{array}{l} \text{Probability that none} \\ \text{of the balls } 1,2,\ldots,n \\ \text{overflow} \end{array} \right\}$$

$$= \sum_{i=1}^{n} i\{P(i-1,m,b) - P(i,m,b)\} + nP(n,m,b)$$

$$= \sum_{i=0}^{n-1} P(i,m,b).$$

$E(n,m,b)$ can be viewed as the area under the plot of $P(n,m,b)$ against n. A good approximation for the purpose of computing $E(n,m,b)$ is to assume that $P(n,m,b)$ is one for $n < n_l$, is zero for $n > n_h$ and that it falls linearly from one to zero as n increases from n_l to n_h, where n_l and n_h are such that $P(n_l,m,b) = 1 - \frac{1}{m}$ and $P(n_h,m,b) = \frac{1}{m}$. Further analysis of the expected cost of finding perfect hashing functions can be found in [9].

7. CONCLUSIONS

We have presented a recurrence relation and algorithm to compute the probability of a random distribution of n balls into m urns each of size b resulting in no overflows. This solves the computational problem of a classical combinatorial extreme-value distribution. The analysis was crucial in the design of a practical and competitive perfect hashing scheme for large external files[7,9]. The results and the techniques presented in this paper also enabled solution of other problems in [6,8,10]. The main problem solved in [10] is to derive an exact probability distribution of the number of balls overflowing, when n balls are randomly tossed into m urns each having a capacity of b balls.

APPENDIX A

Algorithm to compute a table of $P(n,m,b)$.

```
procedure pnmb (parameters: mmax, b)
b, mmax : integer ;
P: array[0..b * mmax, 1..mmax] of real;      {P[i,j] stores P(i,j,b)}

begin
      m, n, deficit : integer;
      term : real;

      {initialize P(i,j,b) = 1.0 for 1 ≤ i ≤ b and 1 ≤ j ≤ mmax}
      for m := 1 to mmax do
            for n := 0 to b do
                  P[n,m] := 1.0;

      for m := 2 to mmax do
            deficit := b;
            term := 1.0 ;      {see the notes below}
            for n := b+1 to m * b do
                  adjust_term(term, m, deficit);
                  P[n,m] := P[n-1,m] - term * P[n-b-1,m-1] ;
                  term = term * (m-1)/m * n/(n-b);
            endloop;
      endloop;
end;

procedure adjust_term(parameters: term, m, deficit)
term : real;
m, deficit : integer;

begin
      while( deficit > 0 and term > machine-epsilon) do
            term := term/m;
            deficit := deficit - 1;
      endloop;
end;
```

The above procedure is a direct implementation of the recurrence relation (4). The initial value of *term* should actually be $1/m^b$. For large values of m and b, this may lead to *term* having

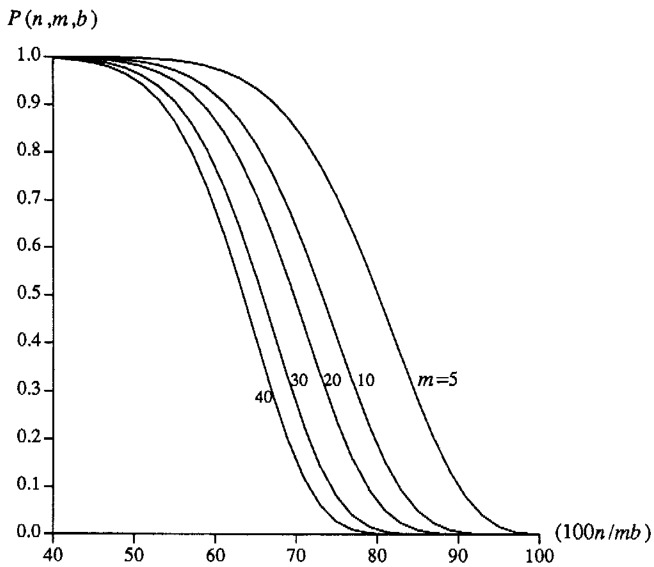

FIG II. Plot of $P(n,m,b)$, Urn capacity (b) : 20.
Number of urns (m) : 5,10,20,30,40

a value too small to be represented by a floating-point number in the computer. However, as the iteration progresses, the value of *term* increases slowly. Considering the range of values of the probabilities at the beginning of the iteration (close to 1.0), if *term* is less than machine-epsilon it is as good as being zero for subtractions (and hence need not be computed accurately). The procedure *adjust_term* handles this problem by not allowing the value of *term* to go very much below machine-epsilon. The incorrect value of *term* at the start of the iteration does not cause any error because in the main procedure the product of *term* and a probability is subtracted from another probability (P[n,m] := P[n−1,m] − term * P[n−b−1, m−1]). (the value of machine-epsilon is a measure of the accuracy of real number arithmetic of the computer. It is adequate to view machine-epsilon as the largest real number such that the addition 1.0 + machine-epsilon gives a sum of precisely 1.0. The same applies for subtraction. Typically machine-epsilon is of the order of 10^{-7} to 10^{-16}.)

The numbers involved are well scaled and the procedure is computationally stable. Round-off errors do not cause any problems unless the value of $P(n,m,b)$ is of the order of machine-epsilon. When $P(n,m,b)$ is very small, of the order of machine-epsilon, the exact values may still be computed using the recurrence relation (6) for $R(n,m,b)$.

Figure II plots the probabilities $P(n,m,b)$ computed using the procedure given above, against n expressed as a percentage of the full capacity mb of the m urns. The higher curves correspond to lower values of m. The graphs indicate that $P(n,m,b)$ drops very rapidly from almost 1.0 to almost 0.0 within a narrow range n. This critical region becomes narrower as b increases and shifts slowly towards zero as the value of m increases. In section 5 we have analyzed the movement of the critical region for increasingly large values of m.

ACKNOWLEDGEMENTS

I wish to thank G.V.Cormack, R.Dubes, P.A.Larson, D.Matthews and J.F.Monahan for their help in the preparation of this paper.

REFERENCES

[1] Barton, D.E. and David, F.N.(1959). *Combinatorial extreme value distributions.* Mathematika, 6, 63 - 76.

[2] Char, B.W., Geddes, K.O., Gonnet, G.H. and Watt, S.M.(1983). *Maple User's Manual, 3rd edition.* Department of Computer Science, University of Waterloo, Research Report CS-83-41.

[3] David, F.N. and Barton, D.E.(1962). *Combinatorial Chance.* London: Griffin.

[4] Feller, W.(1968). *An Introduction to Probability Theory and its Applications.* Vol. 1. New York: John Wiley.

[5] Kolchin, V.F., Sevast'yanov, B.A. and Chistyakov, V.P.(1978). *Random Allocations.* New York: John Wiley and sons.

[6] Larson, P.-A.(1983). *Analysis of uniform hashing.* Journal of the ACM, 30, 4, 805 - 819.

[7] Larson, P.-A. and Ramakrishna, M.V.(1985). *External perfect hashing,* Proc. ACM-SIGMOD Intern'l Conf. on Management of Data, (Austin), 190 - 200.

[8] Norton, R.M. and Yeager, D.P.(1985). *A probability model for overflow sufficiency in small hash tables.* Comm. of the ACM, 28, 10, 1068 - 1075.

[9] Ramakrishna, M.V.(1986). *Perfect hashing for external files.* Ph.D. Thesis, Department of Computer Science, University of Waterloo, Research Report CS-86-25.

[10] Ramakrishna, M.V.(1986). *Overflow sufficiency in small hash tables.* Submitted for publication.

Received by Editorial Board member.

Recommended by John F. Monahan, North Carolina State University, Raleigh, NC.