J. A. van der Pool

Optimum Storage Allocation for Initial Loading of a File

Abstract: When a file is loaded into a direct access storage device using key-to-address transformations, the number and size of storage blocks can be selected. In this study, a selection that minimizes the combined cost of storage space and accesses to the storage device is determined for the case where no record additions or deletions occur after loading.

The analysis is based on the assumption that for a given set of keys, a transformation exists that gives a uniform probability distribution over the available addresses. Under this assumption, formulas are derived for the average number of overflow records and for the average number of accesses required to retrieve a record.

Given these formulas, the costs are expressed as a function of storage used, number of accesses, cost per unit of storage, and cost per access. Minima are computed for a range of block sizes and operational conditions. The results seem to indicate that current file design practices are abundant with storage space.

Finally, the results are condensed in an easy to use approximate formula.

Introduction

Computerized data are organized as files. A file is a collection of data records; in our case, each record is uniquely identified by a key.

In automatic data processing, the data items are frequently stored on a direct access storage device (DASD). A general situation in computer programs using a DASD is the following: a key is shown to the program and the record identified by that key is to be retrieved from the storage device.

A fast retrieval from a DASD is possible if the relationship between the key and the address of the record in the device is known. If such a relationship is not established, a search through the file must be made.

Two methods for establishing the relationship between key and address are in common usage (for a more extensive treatment see [1,2]):

1. An index is maintained in the same or another storage device. For every key, the corresponding address is listed in the index. With a suitable organization, a search through the index is much faster than a search through the file itself.
2. An algorithm defines a function $f:K \rightarrow A$ with the set K of possible keys as domain and the set A of available addresses as codomain. Such a function is often called a key-to-address transformation (KAT). Other names are hashing, hash coding, or scatter storage technique [3].

In this paper, the second method of file organization is considered. Many different KAT algorithms have been described in the literature. A systematic comparative evaluation of several algorithms has been made by Lum, et al. [4] using simulation methods. It appears that the results of different transformations vary widely and are highly dependent upon the characteristics of the set of keys.

In the ideal situation, a KAT should assign an equal number of records to each of the available addresses. Often this goal is pursued, but clearly not reached, by giving the KAT the property of "randomizing." For an analysis of the rationale of this approach see Lum, et al. [4].

It is assumed here that for a given set of keys we are able to find a perfect random transformation. By this we mean that the outcome of the assignment of an address by this KAT is a random variable with equal probability for each of the values it can assume (i.e., having a uniform probability distribution). Under this assumption, formulas are derived for the number of overflow records and the average number of accesses per record. Some of the results of [4] are then compared with the numbers computed from the formulas.

A location in the direct access storage device is designated by an address of the set A, and it may provide room for one or more records. In the latter case, several records are combined in one storage block. We use the name "bucket" for such a block. The bucket size s is the maximum number of records that can be contained in a bucket. The collection of b buckets provided in the

DASD for storing the file is called the primary storage area.

Because of the random nature of the KAT, more than s record keys may be mapped into the same address. There are different methods for storing the records that exceed the capacity of the bucket. Here, we consider the method of using for all buckets a common separate overflow area in the same or another DASD. The overflow records belonging to a certain bucket are organized in a chain stored in the overflow area.

If a record is stored in the primary storage area, a single access to the DASD is sufficient. If the record is in the overflow area, one or more additional accesses are required. In the latter case, the record is found by following pointers from the bucket to the first record, and then from record to record through the overflow area. When this method is implemented for a specific application, the implementer can choose both b and s. If he takes the total capacity bs large relative to the number of records, few additional accesses are required, but the storage cost is high. On the other hand, a low capacity results in high cost for accesses to the overflow storage device.

In this paper we formulate a cost function for retrieving records. Minima are computed for different values of the bucket size and different operational considerations. The file is assumed to be static; that is, no deletions or additions of records occur.

Use of the Poisson Distribution

We consider n records with distinct keys mapped by means of key-to-address transformations into a primary storage area divided into b buckets with a capacity of s records each. If more than s records are assigned to a bucket, the excess records are stored in a separate overflow area common to all buckets. For each bucket with overflow records, there is in the overflow area a chain of overflow records with address pointers.

Assuming that we have an equal probability of assigning a record to any of the available buckets, the number of records r assigned to a bucket will have a binominal probability distribution with parameters $1/b$ and n:

$$b\left(r;n,\frac{1}{b}\right)=\binom{n}{r}\left(\frac{1}{b}\right)^{r}\left(1-\frac{1}{b}\right)^{n-r}$$

We are interested in cases where both n and b are large, and where the average number of records assigned to a bucket $n/b = m$ is nearly equal to the bucket size s. The binominal distribution is then very close to the Poisson distribution with parameter m (see, e.g., [5], pp. 142 ff.):

$$P(r)=\frac{e^{-m}m^{r}}{r!}.$$

Here, $P(r)$ is the probability that r records are assigned to a bucket. We further introduce $Q(r)=\sum_{i=r}^{\infty}P(i)$. The formulas for overflow and accesses will be expressed in these quantities. Since tables are available [6], the formulas can be used for hand as well as for machine computation.

Overflow

The frequency function $O(i)$ of the number i of records assigned to a bucket in excess of its capacity s is:

$$O(0)=\sum_{r=0}^{s}P(r)\,,$$

$$O(i)=P(s+i)\text{ for }i\geq 1\,.$$

The mean of this distribution is:

$$\bar{i}(m,s)=\sum_{i=1}^{\infty}iO(i)=\sum_{i=1}^{\infty}iP(s+i)=\sum_{r=s+1}^{\infty}(r-s)P(r)\,,\quad(1)$$

which can be expressed as:

$$\bar{i}(m,s)=mQ(s)-sQ(s+1)$$

or

$$\bar{i}(m,s)=sP(s)-(s-m)Q(s)\,.$$

We can easily see that for $s = 1$

$$\bar{i}(m,1)=m+e^{-m}-1\,.$$

This result was reported earlier by Morris [3].

If there is no ambiguity, we will write $\bar{i}$ instead of $\bar{i}(m,s)$. A table of values for overflow is given by Buchholtz [2], who computed the values by a recursive procedure, but did not give an explicit formula.

If we apply the formula for $s = 0$, we get $\bar{i}(m,0) = m$. This is a correct result since, in this case, all records are in the overflow area.

Additional accesses

Records that require i additional accesses occur when a bucket has at least $(s + i)$ records assigned to it by the key-to-address transformation. To each such bucket there belongs just one such record. This, the average number of records that requires exactly i additional accesses, is:

$$n_i=b\{P(s+i)+P(s+i+1)+\cdots\}=b\sum_{r=s+i}^{\infty}P(r)\,.$$

The average over the collection of records is:

$$\bar{a}(m,s)=\frac{1}{n}\sum_{i=1}^{\infty}n_i i$$

$$\bar{a}(m,s)=\frac{b}{n}\sum_{i=1}^{\infty}i\sum_{r=s+i}^{\infty}P(r)=\frac{1}{m}\sum_{i=1}^{\infty}i\sum_{r=s+i}^{\infty}P(r)$$

$$=\frac{1}{2m}\sum_{r=s+1}^{\infty}(r-s)\,(r-s+1)P(r)\,.$$

Table 1 Overflow for $s = 1$ and various load factors.

Load factor l	Poisson parameter $m = ls$	Mean overflow $\bar{i}$	Percentage overflow $\frac{\bar{i}}{m} \cdot 100$	Percentage overflow Overall*	Percentage overflow Division*	Utilization primary area %
0.50	0.50	0.1065	21.3	23	17	39.3
0.55	0.55	0.1269	23.1	25	21	
0.60	0.60	0.1488	24.8	27	20	45.1
0.65	0.65	0.1720	26.4	28	27	
0.70	0.70	0.1966	28.1	30	25	50.3
0.75	0.75	0.2224	29.6	31	25	
0.80	0.80	0.2493	31.2	33	28	55.1
0.85	0.85	0.2774	32.6	34	30	
0.90	0.90	0.3066	34.1	37	29	59.3
0.95	0.95	0.3367	35.4	36	33	
1.00	1.00	0.3679	36.8			63.2

*Results from simulation as described in [4].

Table 2 Overflow for $s = 10$ and various load factors.

Load factor l	Poisson parameter $m = ls$	Mean overflow $\bar{i}$	Percentage overflow $\frac{\bar{i}}{m} \cdot 100$	Percentage overflow Overall*	Percentage overflow Division*	Utilization primary area %
0.50	5.0	0.0222	0.4	1	0	49.8
0.55	5.5	0.0433	0.9	1	1	
0.60	6.0	0.0773	1.3	3	1	59.2
0.65	6.5	0.1286	2.1	2	1	
0.70	7.0	0.2013	2.9	3	2	68.0
0.75	7.5	0.2993	4.1	4	3	
0.80	8.0	0.4259	5.3	6	4	75.7
0.85	8.5	0.5833	7.0	8	7	
0.90	9.0	0.7732	8.6	10	7	82.3
0.95	9.5	0.9959	10.6	11	8	
1.00	10.0	1.2511	12.5			87.9

*Results from simulations as described in [4].

If during the accessing phase records are equally demanded, the average number of additional accesses is also $\bar{a}$.

In terms of $P(s)$ and $Q(s)$, we find:

$$\bar{a}(m,s) = \frac{1}{2m}\left[\{(m-s+1)^2 + (s-1)\}\, Q(s) + s(m-s+1)P(s)\right].$$

If there is no possibility of ambiguity, we will also write $\bar{a}$ instead of $\bar{a}(m,s)$.

Another way of expressing the result is:

$$\bar{a}(m,s) = \frac{1}{2}\left\{\bar{i}(m,s-1) - (s-1)\,\frac{\bar{i}(m,s)}{m}\right\}.$$

We note a special case of this general expression:

$$\bar{a} = \frac{m}{2} \qquad \text{for } s = 1.$$

This result was reported earlier by Morris [3].

Comparison with actual mappings

Reference 4 presents results for eight different key-to-address transformations applied to eight files. The load factor and bucket size were varied over a wide range. The load factor l is the ratio of the number of records n and the capacity $b \cdot s$ of primary storage.

Since $l = n/bs = m/s$ or $m = ls$, the parameter of the Poisson distribution is determined by load factor and bucket size.

For the cases $s = 1$ and $s = 10$ and for load factors 0.5 to 0.95 at intervals of 0.05, the values computed by the formulas can be compared with the results found in [4]. Tables 1 and 2 give the comparison for the overflow area. Listed are the "overall" average of all eight key-to-address transformations and the result of mapping by the "division" method. The results are plotted in Figs. 1 and 2.

The tables give some additional information. Since the n records are distributed over the primary and overflow areas, the utilization is clearly lower than the load factor. The utilization is $[(ls - \bar{i})/s]\ 100\%$ of the primary area.

A comparison between the average number of additional accesses from formula and experiment is made in Figs. 3 and 4 for bucket sizes $s = 1$ and $s = 10$, respectively.

In Figs. 1–4, we see that the curves for the "overall" average that is analyzed in [4] lie above those computed by the formulas. This means that the performance resulting from the use of those transformations is, for the files studied, not as good as would be expected under the assumption of a uniform distribution over the address space.

On the other hand, we see that a particular transformation may give better performance than expected from random distribution. For the files studied in [4], the division method leads to a small overflow area and fewer additional accesses. This seems to suggest the following approach in systems design. The formulas derived in this report can be used in a first investigation of storage space and timing studies. When an actual key-to-address transformation is selected and used, the performance can be compared with the previous analytical results. If the selected key-to-address transformation gives more overflow or more additional accesses, it has an undesirable interaction with the specific set of keys, and other transformations or transform parameters should be tried.

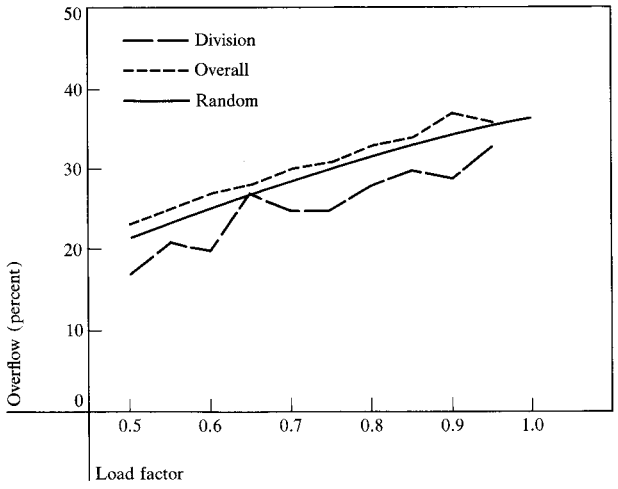
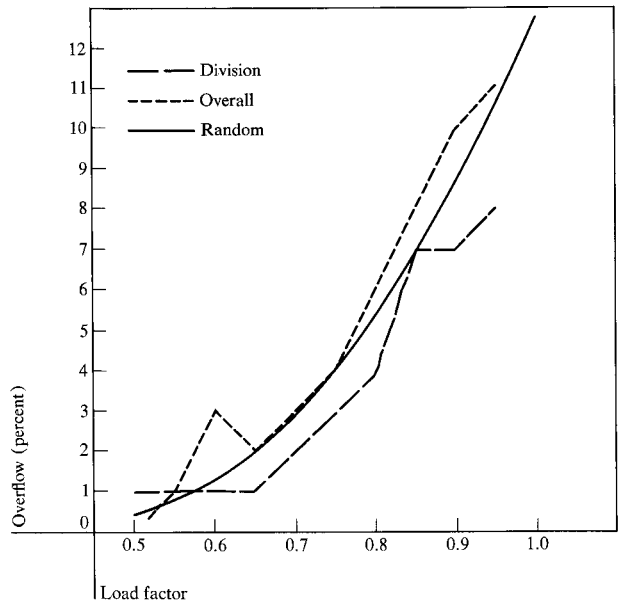


Figure 1 Percentage overflow for $s = 1$.

Figure 2 Percentage overflow for $s = 10$.



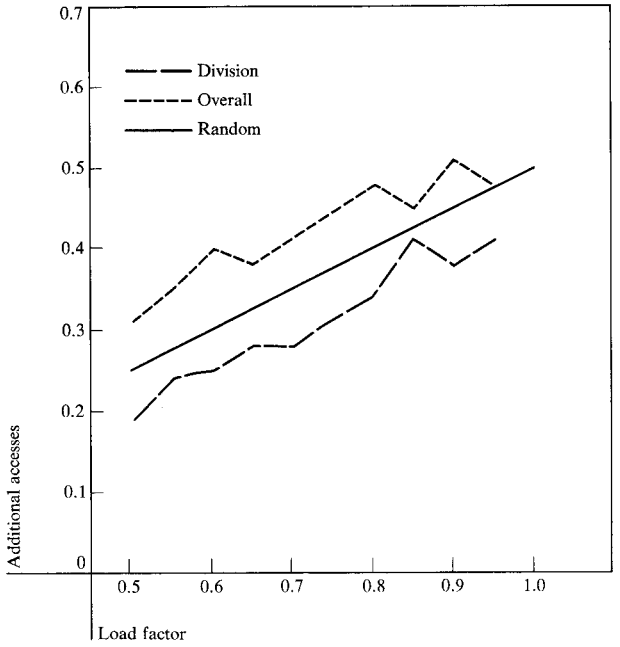
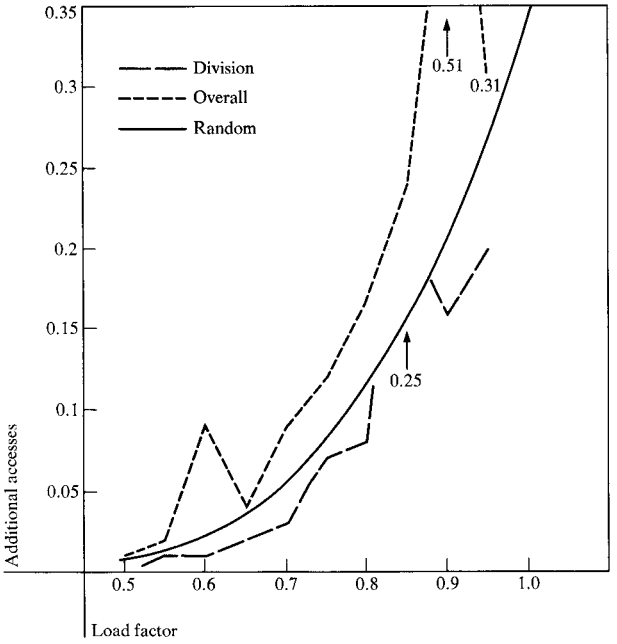


Figure 3 Additional accesses for $s = 1$.

Figure 4 Additional accesses for $s = 10$.



Minimum cost allocation

With the formulas developed in the previous paragraphs, and if the designer of the file can attribute fixed constant costs to access and to units of storage space, we can now proceed to investigate a minimum cost storage allocation.

We will use the following symbols:

$\bar{t} = b(s + \bar{i})$ total average storage used for the files;
c_s cost for storing one record during a unit of time;
α file activity, i.e., the fraction of records required during a unit of time;
c_p cost of an access to the primary area;
c_a cost of an additional access.

If we neglect other costs, such as the transmission from auxiliary to main storage and the cost of main storage for holding a physical record, the total cost per time period is:

$$C(b,s;\alpha) = \bar{t} \cdot c_s + \alpha n c_p + \alpha n \bar{a} \cdot c_a .$$

The term $\alpha n c_p$ expresses the cost for accesses to the primary storage area. It is assumed that c_p is independent of b and s. Further we make a simplification by disregarding the maximum track length in physical devices. We normalize the variable cost by dividing by the nominal cost for storing n records. The variable relative cost function is:

$$D(b,s;\alpha) = \frac{\bar{t} \cdot c_s + \alpha n \bar{a} \cdot c_a}{n c_s} = \frac{s + \bar{t}}{m} + \gamma \bar{a}$$

in which $\gamma = \alpha \frac{c_a}{c_s}$.

It can be shown that for each fixed s there is just one value of m that makes $D(b,s;\alpha)$ minimal. The location and the value of the minimum can be determined with straightforward analytic and numerical methods.

Using APL [9], programs were developed to compute the value of m that gives minimum cost. The computations were made over a wide range of values of the parameter γ and for different values of the bucket size s. The results are shown in Table 3. For each minimum, a number of related quantities have been computed. The load factor is the ratio between the number of records loaded and the number of record positions in the primary storage area, thus n/bs or m/s. The overflow factor is the ratio between the number of records in the overflow area and the total number of records loaded, thus $\bar{o}/n$ or $\bar{t}/m$. Additional accesses are the average number of additional accesses that have to be made to find the addressed record, thus $\bar{a}$. Finally, the minimum relative costs are listed in the last column.

The application factor γ

In the computation of the minimum, the parameter γ plays an important role. It is determined from the file application and by the environment in which the application runs. A high γ means that either or both: 1) number of accesses and 2) cost of additional accesses are given a high value. An estimate for the activity α in $\gamma = \alpha\, c_a/c_s$ can be made in a straightforward manner from the given application.

To get an idea of the magnitude of the estimate for c_a/c_s, let us assume that we store 250,000 records in a disk storage device. Further, let us assume that the time for an additional access is 100 ms. If we suppose an eight-hour work day as the unit of time and, at one extreme, that an additional access ties up only the disk device, we can take:

$$\frac{c_a}{c_s} = \frac{[0.1/(8 \times 3600)] \text{ rent device per day}}{(1/250{,}000) \text{ rent device per day}} = 0.9\,.$$

Cost of additional accesses is evaluated too highly in this case, because cost of the storage device has already been accounted for in the cost of storing records. A more accurate estimate is the cost of interference in the rest of the computer system.

With a low activity, say $\alpha = 0.01$, then $\gamma = 0.01$. This is the lowest value we used in our calculations. Values of over 1 will be very rare under the specified conditions.

The above estimate is approximately valid in a multiprogramming environment. In this case, the rest of the computer system is used while the additional accesses go on. If the file application under consideration is the only application in the system, then we must reckon with the fact that the whole system is tied up during the additional access. Then, we have to put "rent system per day" instead of "rent device per day" in the numerator. The value of γ may then be up to one order of magnitude larger.

Another consideration may be response time. If response time is a prime consideration, a higher value can be assigned to γ. The average number of additional accesses specified in Table 3, together with the file activity, may be helpful in estimating average response times.

Analysis of results

From Table 1, we see that for each value of γ, m increases monotonically with s. The relation is almost linear (see Fig. 5).

The minimum cost decreases when we increase the bucket size s. As a matter of fact, practical considerations not included in this model prevent us from making s very large. Interfering factors are mainly:

1. Direct access devices have tracks of finite length. We cannot usefully make buckets larger than tracks. Similarly, storage waste occurs if one or more buckets do not fill a track completely. The effects of these factors have been studied [7]. Since technology is providing devices with longer tracks, the influence of these factors is decreasing.
2. In reading and writing, the contents of the bucket must be contained in the working storage of the computer. Very large buckets put heavy demand on main storage. Here again technology is providing larger memories allowing for larger buckets.

Quite as expected, m and thus the load factor for a given s increase when γ decreases. The author is under the impression that the values of the load factor that are found to give minimum cost are much higher than those used in practical applications. In a life insurance company, for instance, γ will not be higher than 0.1. Using a bucket size of 10, the optimal load factor would be over 1.2. It seems unlikely that many companies, at present, design their files to have more than 20 percent of the records in the overflow area. Another example is in the section on indirectly addressed files in [1], page 42, "an initial packing (loading) factor of 80–85 percent is

Table 3 Minimum cost computations.

Bucket size	γ	m	*Load factor*	*Overflow factor*	*Additional accesses*	*Minimum cost*
1	2	0.883	0.883	0.336	0.441	2.351
	1	1.163	1.163	0.409	0.581	1.850
	0.5	1.496	1.496	0.481	0.748	1.524
	0.1	2.445	2.445	0.626	1.222	1.158
	0.05	2.914	2.914	0.675	1.457	1.091
	0.01	4.104	4.104	0.760	2.052	1.025
2	2	1.586	0.793	0.202	0.294	2.052
	1	1.977	0.988	0.267	0.424	1.703
	0.5	2.428	1.214	0.337	0.589	1.456
	0.1	3.664	1.832	0.494	1.098	1.149
	0.05	4.255	2.127	0.551	1.359	1.089
	0.01	5.705	2.853	0.654	2.027	1.025
3	2	2.317	0.772	0.144	0.227	1.893
	1	2.795	0.932	0.200	0.344	1.617
	0.5	3.339	1.113	0.264	0.501	1.412
	0.1	4.797	1.599	0.416	1.011	1.143
	0.05	5.481	1.827	0.475	1.282	1.086
	0.01	7.134	2.378	0.584	1.987	1.025
4	2	3.068	0.767	0.112	0.188	1.791
	1	3.621	0.905	0.161	0.294	1.560
	0.5	4.244	1.061	0.218	0.442	1.382
	0.1	5.892	1.473	0.363	0.945	1.137
	0.05	6.656	1.664	0.422	1.220	1.084
	0.01	8.481	2.120	0.533	1.946	1.024
5	2	3.834	0.767	0.092	0.162	1.719
	1	4.454	0.891	0.134	0.260	1.517
	0.5	5.147	1.029	0.187	0.400	1.358
	0.1	6.963	1.393	0.325	0.893	1.132
	0.05	7.799	1.560	0.382	1.170	1.082
	0.01	9.778	1.956	0.494	1.910	1.024
10	2	7.813	0.781	0.048	0.102	1.531
	1	8.697	0.870	0.075	0.175	1.400
	0.5	9.669	0.967	0.112	0.287	1.289
	0.1	12.158	1.216	0.221	0.734	1.117
	0.05	13.278	1.328	0.271	1.005	1.074
	0.01	15.876	1.588	0.376	1.768	1.023
20	2	16.156	0.808	0.025	0.064	1.391
	1	17.417	0.871	0.041	0.117	1.306
	0.5	18.787	0.939	0.064	0.203	1.230
	0.1	22.245	1.112	0.143	0.588	1.101
	0.05	23.781	1.189	0.183	0.841	1.066
	0.01	27.285	1.364	0.273	1.598	1.022
40	2	33.518	0.838	0.012	0.041	1.288
	1	35.308	0.883	0.022	0.078	1.233
	0.5	37.239	0.931	0.035	0.142	1.181
	0.1	42.086	1.052	0.089	0.460	1.085
	0.05	44.224	1.106	0.119	0.688	1.057
	0.01	49.054	1.226	0.190	1.416	1.020

suggested." It is not clear on what this figure is based, nor are the influence of activity and the cost of accesses pointed out.

One can wonder whether inadequacies in our model cause the discrepancy between theory and practice. The following considerations, however, indicate that even higher load factors may be appropriate. Underlying the formula used for additional access is the assumption that all records are in equal demand. In many practical situations, this assumption does not hold. More often, 20 percent of the records will have 80 percent of the demand. While this will seldom be known in advance (e.g., sales

statistics) or will be found during the application, one can later load the file in the order of decreasing frequency of demand. The records with the highest demand rate will then go into the primary storage area, thus reducing the average number of additional accesses considerably. Heising [8] estimates that for $s = 1$ and a load factor of 1, the reduction in additional accesses is from 0.5 for equal demand to 0.12 when a "20 – 80 percent rule" holds.

It appears from [4] that in actual situations key-to-address transformations may be found that give a better performance than the perfect random transformation; thus, again, fewer additional accesses will be required. The designer should however be aware of the irregularities that may occur in practical situations as illustrated by the anomolous behavior for particular load factors in Fig. 4.

Both considerations indicate that the load factors found in this report are lower than may be appropriate in practical situations. A factor that counteracts higher load factors is the way in which the cost for storage is accounted for. In the cost function, it was assumed that the cost of primary and overflow storage is the same. In practical situations the cost per record in overflow storage will be somewhat larger because more space per record is needed. Factors contributing to this extra space requirement include the fact that overflow records are not blocked, space beyond that required for the average number of overflow records must be provided and space for pointers is included. It seems that in current practice many system designers are using load factors that are lower than the optimum.

If we want to make a very general rule of thumb, then it is more appropriate to suggest a load factor of 1 instead of 0.8 to 0.85. Only for high values of γ are the latter more appropriate.

The values of the minimum cost are graphed in Fig. 6. We can see that for low values of γ the minimum is very insensitive to the bucket size. In practice, the dependence is greater. The number of characters on a track is dependent on the number of blocks (buckets) that are stored on the track. This nonlinearity is not represented in the model used.

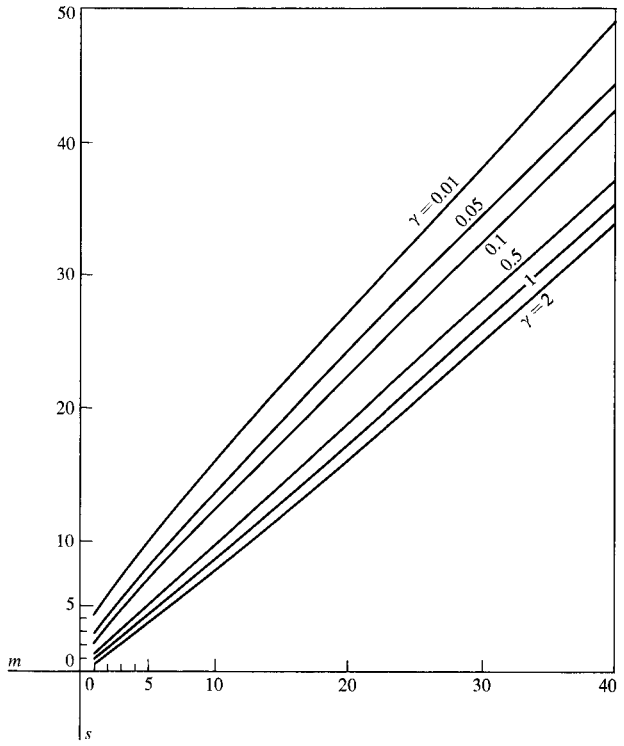
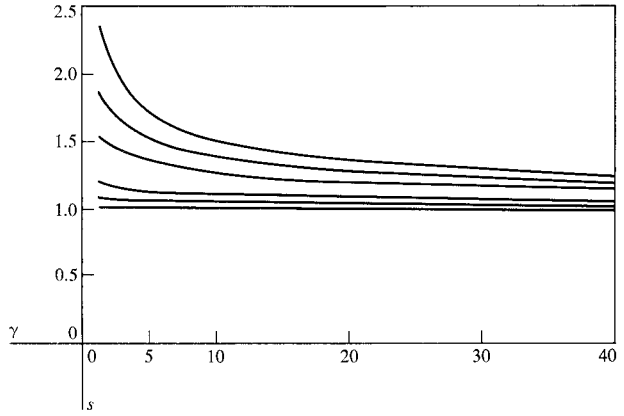


Figure 5 Minimum cost m as a function of s.

Figure 6 Minimum cost as a function of s for various γ.



Practical approximation

As was pointed out, the functional relationship $m = f(s;\gamma)$ between the optimum value of m and s is nearly linear. This gives us the opportunity to condense the results found into a simple design rule.

First, straight lines $\hat{m} = p(\gamma) + g(\gamma) \cdot s$ are fitted to the curves pictured in Fig. 5 by means of least squares. In Fig. 7, the values of $p(\gamma)$ and $q(\gamma)$ are plotted. This graph suggests an approximately linear function for q and a logarithmic relationship between p and γ. Again, by least squares, curves were fitted to $p(\gamma)$ and $q(\gamma)$.

This now gives the following design rule:

1. Make an estimate of $\gamma = \alpha \frac{c_a}{c_s}$,
2. The optimum value of the load factor $l = \frac{m}{s}$ for every s is:

$$l = \frac{p}{s} + q$$

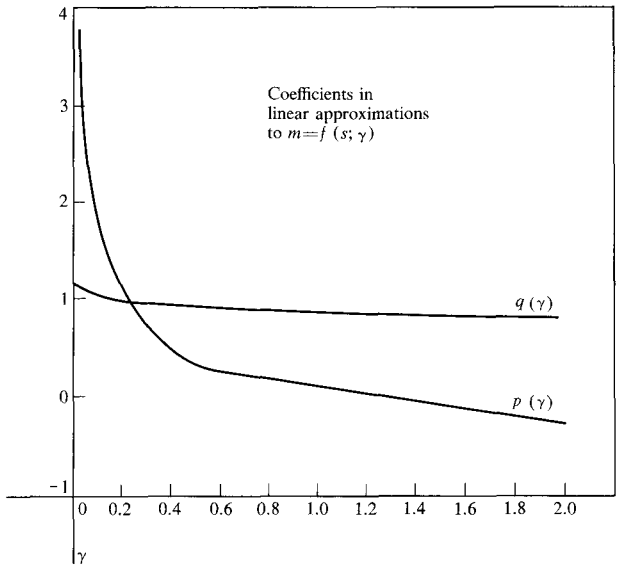


Figure 7 Values of $q(\gamma)$ and $p(\gamma)$.

in which

$$p = 0.13 - 0.76 \cdot ln\gamma$$

$$q = 1.05 - 0.13 \cdot \gamma .$$

The discrepancies using this approximation are given in Table 4. As we see, this approximation is not usable for small s and large γ but it is otherwise acceptable for all practical purposes.

This formula allows the system designer to select bucket size with due consideration to properties of the physical device, and then to make in an easy way an estimate of the optimal load factor belonging to this s.

Acknowledgments

The author wishes to express his gratitude to M. E. Senko and S. P. Ghosh for their suggestions, discussions, and encouragement.

Table 4 Excess in percent of minimum cost if the approximation $m = p(\gamma) + q(\gamma) \cdot s$ to $m = f(s;\gamma)$ is used.

	γ					
s	2	1	0.5	0.1	0.05	0.01
1	32.2	0.4	0.3	0.6	0.4	0.1
2	6.2	0.0	0.3	0.2	0.1	0.0
3	2.5	0.1	0.5	0.1	0.0	0.0
4	1.3	0.3	0.6	0.0	0.0	0.1
5	0.8	0.5	0.7	0.0	0.0	0.2
10	0.4	1.0	1.2	0.0	0.1	0.4
20	0.8	1.3	1.8	0.0	0.1	0.6
40	2.3	1.1	2.5	0.2	0.0	0.5

References

1. *Introduction to IBM System/360 Direct Access Storage Devices and Organization Methods*, IBM Student Text, Form No. GC 20-1649.
2. W. Buchholz, "File Organization and Addressing," *IBM Systems Journal* **2,** 86 (1963).
3. R. Morris, "Scatter Storage Techniques," *Communications of the ACM* **11,** No. 1, pp. 38–44 (January 1968).
4. V. Y. Lum, P. S. T. Yuen, and M. Dodd, "Key-to-Address Transform Techniques: A Fundamental Performance Study on Large Existing Formatted Files," *Communications of the ACM* **14,** No. 4 (April 1971).
5. W. Feller, *An Introduction to Probability Theory and Its Applications*. John Wiley and Sons, Inc., New York.
6. E. C. Molina, *Poisson's Exponential Binomial Limit*, D. van Nostrand Co. (1942).
7. "File Design Handbook", Final Report, Air Force Contract AF30602.69-C-0100, (November 1969).
8. W. P. Heising, "Note on Random Addressing Techniques," *IBM Systems Journal* **2,** 112 (1963).
9. S. Pakin, *APL/360 Reference Manual*, Science Research Associates, Chicago.

Received January 10, 1972

The author is located at the head office of IBM Nederland N.V., Amsterdam, The Netherlands.