Operating Systems
R. S. Gaines Editor

Optimal Storage Allocation for Serial Files

Haim Mendelson, Joseph S. Pliskin, and Uri Yechiali
Tel Aviv University

A computer system uses several serial files. The files reside on a direct-access storage device in which storage space is limited. Records are added to the files either by jobs in batch processing mode, or by on-line transactions. Each transaction (or job) generates a demand vector which designates the space required in each file for record addition. Whenever one file runs out of space, the system must be reorganized. This paper considers several criteria for best allocating storage space to the files.

Key Words and Phrases: serial files, storage allocation, reorganization, partitioned dataset
CR Categories: 3.5, 3.7, 4.33, 4.6

1. Introduction

External storage allocation is one of the final steps in the physical design of a computer application. Several studies have considered this problem [1, 2, 7, 8]. They examined the tradeoff between efficient storage allocation and system performance (some considered additional factors as well). Each evaluated some specified performance measures for a given file organization scheme.

In this study we consider a computer application based on a few interrelated files residing on direct-access media. Records are added to a file sequentially, and are stored at the end of the dataset. (Functions that do not consume storage space—e.g. retrieval or replacement of a record—are not necessarily processed sequentially.) We term such a dataset a *serial* file. Serial files are widely used in data processing applications. The most common form is a sequential dataset, where records are always processed in the sequence in which the file is constructed. Other examples are a partitioned dataset (analyzed in detail in Section 5), and the overflow area of some index-sequential file organization schemes. Serial overflow files are also used with some types of random datasets.

Since the storage space allocation for each file is finite, the system has to be reorganized from time to time. The reorganization procedure depends on the specific structure of the system; we may assume that after reorganization, the serial files are empty and the process of record addition may be initiated. The problem is then how to best allocate a given total amount of storage space S among the files. This problem was partially attacked by the authors [3], where the criterion for allocation was the maximization of the expected time until reorganization. In the present work we develop the optimal allocation for a "reliability" criterion, and compare the optimal allocation rules for the various criteria.

We emphasize the need for simple allocation rules to close the gap between theory and application. As it turns out, the optimal allocation rules derived in this work are simple enough to be easily implemented.

The model and the various optimization criteria are introduced in Section 2. Allocation that maximizes the expected time until reorganization is discussed in Section 3, while in Section 4 we find the allocation that maximizes the probability that the system can process a specified number of transactions before a reorganization is required (this is the "reliability" criterion). In Section 5 we present two applications and numerical examples for the results developed in the preceding sections.

2. Criteria for Optimization

Consider a computer application that uses n interrelated serial files. The files reside on a direct-access storage device, where storage space is limited. Records are added to the files from time to time, either by jobs in a batch processing mode, or by on-line transactions. Transactions[1] arrive into the system at instants $0 < t^1 < t^2 < \ldots < t^k < \ldots$, where the interarrival times, $\tau^k = t^k - t^{k-1}$, are independent, identically distributed (iid) random variables. We assume the distribution of each τ^k ($k = 1, 2, 3, \ldots$) to be identical to that of a nonnegative random variable τ ($\tau^k \sim \tau$) possessing a finite mean $E\tau$. Each transaction generates a demand vector which designates the space required in each file for record addition. The kth demand is an n-dimensional vector $\mathbf{V}^k = (V_1^k, V_2^k, \ldots, V_n^k)$, where V_i^k is the amount of storage space required in the ith file due to the processing of the

[1] Although we have chosen to adopt the teleprocessing terminology, our results are valid for batch processing as well.

Authors' address: H. Mendelson and U. Yechiali, Department of Statistics, J.S. Pliskin, Faculty of Management, Tel Aviv University, Tel Aviv, Israel.

kth transaction. $\{\mathbf{V}^k\}_{k=1}^{\infty}$ are independent random variables identically distributed as a random variable $\mathbf{V}(\mathbf{V}^k \sim \mathbf{V})$ whose distribution function is $F(\mathbf{v}) = P\{\mathbf{V} \leq \mathbf{v}\}$. The vector $\mathbf{V}$ is nonnegative with finite mean $E(\mathbf{V}) = \mu = (\mu_1, \mu_2, \ldots, \mu_n)$.

Let S be the amount of storage space available for allocation. At time $t^0 = 0$, S is allocated to the files. Let $x_i \geq 0$ be the amount of space allotted to the ith file ($i = 1, 2, \ldots, n$). Let

$$D = \{\mathbf{x} | \mathbf{x} \in R^n, \quad \mathbf{x} \geq \mathbf{0}, \quad \sum_{i=1}^{n} x_i \leq S\}$$

denote the set of all feasible allocation vectors. Let $\mathbf{S}^k = (S_1^k, S_2^k, \ldots, S_n^k)$ denote the vector of total demand for storage up to time t^k, where the cumulative demand of file i is $S_i^k = \sum_{j=1}^{k} V_i^j$.

The system can operate as long as none of the files runs out of space—that is, as long as $\mathbf{S}^k \leq \mathbf{x}$. Whenever one file runs out of space, the system must be reorganized. We say that the system fails at the first moment t^k when $S_i^k > x_i$ for some file i—that is, when it has to be reorganized. Hence the system's lifetime until failure, $T(\mathbf{x})$, is given by

$$T(\mathbf{x}) = \min\{t^k | S_i^k > x_i, \quad \text{for some } i\}.$$

The number of transactions the system is capable of processing without reorganization is given by

$$N(\mathbf{x}) = \max\{k | \mathbf{S}^k \leq \mathbf{x}\}.$$

The random variables $T(\mathbf{x})$ and $N(\mathbf{x})$ are related by the formula

$$T(\mathbf{x}) = \sum_{k=1}^{N(\mathbf{x})+1} \tau^k. \tag{1}$$

Equation (1) simply states that the lifetime of the system equals the sum of interarrival times until failure.

The problem is to chose an allocation vector $\mathbf{x} \in D$ satisfying some optimality criterion. In this paper we consider and compare three criteria for optimality:

(i) Maximization of the expected system lifetime: $\max_{\mathbf{x} \in D} ET(\mathbf{x})$.
(ii) Maximization of the expected number of transactions processed until reorganization: $\max_{\mathbf{x} \in D} EN(\mathbf{x})$.
(iii) Maximization of the probability that the system can process at least M transactions without failure (M is some prespecified number).

The first two criteria are cost oriented, and are typical of batch-processed applications. By maximizing the expected time between reorganizations, the reorganization cost per unit of time is minimized.[2] Hence, an appropriate allotment could decrease average operating costs. Similarly, the second criterion leads to the minimization of the expected reorganization cost per job (or transaction) processed.

The viewpoint of a decision maker adopting the third criterion, which we call the *reliability* criterion, is different. The objective here is to minimize the probability of system failure. Such an objective characterizes teleprocessing applications, in which the cost of a failure is high. For a system that has been designed to process up to M transactions, the reliability function, representing the probability of meeting specifications, is equal to $P\{N(\mathbf{x}) \geq M\}$.

In the sequel, we make use of the fact that the number of transactions processed by a computer system until reorganization is usually very large. Thus, the parameter M defined in the reliability criterion is assumed to be very large. In the analysis of expected value criteria, we similarly assume that $S \gg \sum_{j=1}^{n} \mu_j$. Hence, asymptotic approximations may be use to yield general, simple results.

The allocation problem can also be regarded as a decision problem under uncertainty. The decision maker chooses an action (i.e. allocation vector) $\mathbf{x} \in D$. Then a random consequence $N(\mathbf{x})$ (or $T(\mathbf{x})$) is obtained. If the von Neumann-Morgenstern axioms are satisfied, a utility function U may be defined over the set of possible consequences so that the expected value of U is maximized. Hence, the problem becomes

$$\max_{\mathbf{x} \in D} EU(N(\mathbf{x})) \quad (\text{or } \max_{\mathbf{x} \in D} EU(T(\mathbf{x}))).$$

It is of interest to find the utility function corresponding to each of the aforementioned criteria. Through the utility function one might gain a better understanding of the preferences of the decision maker.

It is obvious that for the first criterion we have $U(T) = T$ (or an increasing linear function of T), whereas for the second we have $U(N) = N$. For the third criterion, the following lemma is relevant:

LEMMA 1. *A decision maker with the step utility function*

$$U(N) = \begin{cases} 0, & N < M \\ 1, & N \geq M \end{cases} \tag{2}$$

should utilize the reliability criterion.

PROOF. Clearly, $EU(N(\mathbf{x})) = P\{N(\mathbf{x}) \geq M\}$. □

Before going into a detailed analysis, we observe that the number of optimization problems can be reduced from three to two as a consequence of the following result:

LEMMA 2.

$$ET(\mathbf{x}) = E[N(\mathbf{x}) + 1] \cdot E\tau$$

PROOF. The lemma readily follows by an application of Wald's theorem [5] to eq. (1). □

Thus an allocation vector $\mathbf{x}^* \in D$ is optimal for criterion (i) if and only if it is optimal for criterion (ii).

[2] Note that for a regenerative reward process [5], the average reward per unit of time tends with probability 1 to the ratio

$$\frac{\text{expected reward per cycle}}{\text{expected length of a regeneration cycle}}$$

3. Allocation by Expected Value

Consider the equivalent criteria of maximizing $EN(\mathbf{x})$ or $ET(\mathbf{x})$. One intuitively expects a proportionate allocation to be optimal. Such an allocation divides the total available storage space, S, according to the ratios of expected demands $\mu_1 : \mu_2 : \ldots : \mu_n$, so that

$$x_i = [\mu_i / \sum_{j=1}^{n} \mu_j] S, \quad i = 1, 2, \ldots, n. \tag{3}$$

It is easy to construct a counterexample showing that a proportionate allocation is not necessarily optimal. Suppose the system consists of two serial files. A transaction creates a variable-length record in each file. The record sizes are independent, and have the same distribution. The record lengths are either 20 or 40 bytes, with equal probabilities. Hence, in our notation, $\mathbf{V} = (V_1, V_2)$, where V_1 and V_2 are iid, with

$$P\{V_i = 20\} = P\{V_i = 40\} = \tfrac{1}{2}, \quad i = 1, 2.$$

If $S = 60$ bytes, then a proportionate allocation yields $x_1 = x_2 = 30$. For this allocation we have

$$N(30, 30) = \begin{cases} 1, & \text{with probability } \frac{1}{4} \\ 0, & \text{with probability } \frac{3}{4}. \end{cases}$$

Hence,

$$EN(30, 30) = \tfrac{1}{4}.$$

On the other hand, if we let $\mathbf{x} = (40, 20)$, we obtain

$$N(40, 20) = \begin{cases} 1, & \text{with probability } \frac{1}{2} \\ 0, & \text{with probability } \frac{1}{2}, \end{cases}$$

with an expected number of transactions

$$EN(40, 20) = \tfrac{1}{2}.$$

Thus, since $EN(40, 20) > EN(30, 30)$, a proportionate allocation is not optimal.

In real life, each file contains numerous records, so that $S \gg \sum_{j=1}^{n} \mu_j$. In this case, the proportionate allocation rule is (asymptotically) optimal, as seen from the following theorem.

THEOREM 1. *If $x_i \gg \mu_i$ for $i = 1, 2, \ldots, n$, then*

$$EN(\mathbf{x}) \approx \min_{i=1,2,\ldots,n} (x_i/\mu_i), \tag{4}$$

and the optimal allocation is given by (3). A proof of Theorem 1 is given in [3].

It follows from Theorem 1 that in real-life situations (where $S \gg \sum_{j=1}^{n} \mu_j$), the system designer has to estimate the expected demand vector $\boldsymbol{\mu}$, and allocate the given space S so that

$$x_1 : x_2 : \ldots : x_n = \mu_1 : \mu_2 : \ldots : \mu_n.$$

This allocation rule satisfies our basic requirements of simplicity and intuitive acceptability. In fact, we believe that it has been frequently used as a heuristic allocation rule.

The marginal value of storage space may be of interest when S can be varied. If S is allocated proportionately, then $x_i/\mu_i = S/\sum_{j=1}^{n} \mu_j$ $(i = 1, 2, \ldots, n)$. Hence, by eq. (4),

$$EN(\mathbf{x}) = S/\sum_{j=1}^{n} \mu_j. \tag{5}$$

That is, $EN(\mathbf{x})$ is a linear function of S. It follows that a unit increase in the total amount of storage space available for allocation increases the expected number of transactions by $1/\sum_{j=1}^{n} \mu_j$, and the expected lifetime of the system by $E\tau/\sum_{j=1}^{n} \mu_j$.

4. A Reliability Criterion

The reliability criterion defined in Section 2 is to maximize the probability of processing at least M transactions before failure. Mathematically, this yields the optimization problem

$$\max_{\mathbf{x} \in D} P\{N(\mathbf{x}) \geq M\}$$

for some given M. The event $\{N(\mathbf{x}) \geq M\}$ is equivalent to the event $\cap_{i=1}^{n} \{\sum_{k=1}^{M} V_i^k \leq x_i\}$ since the system can process at least M transactions if and only if the storage space required for each file does not exceed its allocated space.

If we invoke the additional assumption that the random variables V_i^k are independent for all $i = 1, 2, \ldots, n$; $k = 1, 2, 3, \ldots$, we obtain

$$P\{N(\mathbf{x}) \geq M\} = P\left\{\bigcap_{i=1}^{n} \left\{\sum_{k=1}^{M} V_i^k \leq x_i\right\}\right\} = \prod_{i=1}^{n} P\left\{\sum_{k=1}^{M} V_i^k \leq x_i\right\}. \tag{6}$$

The vector $\mathbf{x}$ that maximizes $P\{N(\mathbf{x}) \geq M\}$ also maximizes the transformation $g(\mathbf{x}) = \log P\{N(\mathbf{x}) \geq M\}$. The optimization problem then becomes (using (6))

$$\max_{\mathbf{x}} g(\mathbf{x}) = \sum_{i=1}^{n} \log P\left\{\sum_{k=1}^{M} V_i^k \leq x_i\right\} \tag{7}$$

$$\text{s.t. } \mathbf{x} \geq \mathbf{0}, \quad \sum_{i=1}^{n} x_i \leq S.$$

As we shall see in the sequel, by the central limit theorem we can assume the probability distribution of $\sum_{k=1}^{M} V_i^k$ $(i = 1, \ldots, n)$ to be absolutely continuous. Let the cumulative distribution function of $\sum_{k=1}^{M} V_i^k$ be F_i, and its density function f_i.

The optimization problem (7) is then a maximization of a continuous function over a closed and bounded set, so that a maximum $\mathbf{x}^*$ exists. Without loss of generality, we can assume that $\sum_{i=1}^{n} x_i^* = S$. (If not, we can always add to x_1^*, for example, the quantity $S - \sum_{i=1}^{n} x_i^*$ to obtain a new maximum without decreasing the value of $g(\mathbf{x})$.) The optimization domain reduces to the simplex

$$D^* = \left\{ \mathbf{x} | \mathbf{x} \geq \mathbf{0}, \quad \sum_{i=1}^{n} x_i = S \right\}.$$

The maximum of $g(\mathbf{x})$ cannot be achieved on the boundary of D^*, where there exists at least one i ($i = 1, \ldots, n$) with $x_i = 0$, because $P\{\sum_{k=1}^{M} V_i^k \leq 0\} = 0$. Thus, the maximum has to be achieved at an interior point of D^*. At an internal extreme point, all the partial derivatives of the Lagrangean

$$L(\mathbf{x}, \lambda) = g(\mathbf{x}) - \lambda\left(\sum_{i=1}^{n} x_i - S\right)$$
$$= \sum_{i=1}^{n} (\log F_i(x_i) - \lambda x_i) + \lambda S$$

must equal zero. This yields the optimality conditions

$$f_i(x_i)/F_i(x_i) = \lambda, \quad i = 1, \ldots, n. \tag{8}$$

We earlier assumed M to take on very large values, so $\sum_{k=1}^{M} V_i^K$ is the sum of a large number of iid random variables distributed as a random variable V_i. If $Var(V_i)$ is finite, it follows from the central limit theorem that the distribution of the random variable

$$\left(\sum_{k=1}^{M} V_i^k - M \cdot \mu_i\right) \Big/ \left(M \, Var(V_i)\right)^{\frac{1}{2}}$$

approaches the standard normal distribution as M approaches infinity. We can then approximate $F_i(\cdot)$ and $f_i(\cdot)$ by

$$f_i(x_i) \approx \varphi(y_i), \quad i = 1, \ldots, n$$

$$F_i(x_i) \approx \Phi(y_i), \quad i = 1, \ldots, n$$

where

$$y_i = [x_i - M \cdot \mu_i]/[M \, Var(V_i)]^{\frac{1}{2}}, \quad i = 1, \ldots, n, \tag{9}$$

$\varphi(y) = (1/\sqrt{2\pi})e^{-y^2/2}$ is the density of the standard normal distribution, and $\Phi(y)$ represents its cumulative distribution function.

The optimality conditions (8) now become

$$\varphi(y_i)/\Phi(y_i) = \lambda, \quad i = 1, \ldots, n. \tag{10}$$

LEMMA 3. *For every $\lambda > 0$ there exists a unique solution y^* to*

$$\varphi(y) - \lambda\Phi(y) = 0. \tag{11}$$

PROOF. Define $h(y) = \varphi(y) - \lambda\Phi(y)$. The function h is differentiable for every y and satisfies

$$h'(y) = \varphi'(y) - \lambda\varphi(y) = -(y + \lambda)\varphi(y).$$

It is easily seen that $h'(y) > 0$ for all $y < -\lambda$, and $h'(y) < 0$ for all $y > -\lambda$. Since $h(-\infty) = 0$, it follows that $h(y) > 0$ for all $y \leq -\lambda$. Hence, a solution to $h(y) = 0$ can exist only in the interval $(-\lambda, \infty)$. But in this interval h is monotonically decreasing, so there exists at most one solution. The existence of a solution y^* results from the fact that $h(\infty) = -\lambda < 0$. □

Lemma 3 implies that for every $\lambda > 0$ there is a unique solution $y_1 = y_2 = \ldots = y_n = y^*$ to the system (10), where y^* is the (unique) solution to (11). Equation (11) establishes a one-to-one relation between y^* and λ. The value of y^* (or λ) can be obtained via the constraint

$$\sum_{j=1}^{n} x_j^* = S. \tag{12}$$

From eq. (9),

$$x_i^* = (M \cdot Var(V_i))^{\frac{1}{2}} \cdot y^* + M \cdot \mu_i, \quad i = 1, \ldots, n.$$

Summing over i,

$$S = y^* \sum_{i=1}^{n} (M \cdot Var(V_i))^{\frac{1}{2}} + M \sum_{i=1}^{n} \mu_i,$$

so

$$y^* = \left(S - M \cdot \sum_{i=1}^{n} \mu_i\right) \Big/ \left(\sum_{i=1}^{n} (M \cdot Var(V_i))^{\frac{1}{2}}\right)$$

and the optimal allocation is

$$x_j^* = M\mu_j + \left(S - M \cdot \sum_{i=1}^{n} \mu_i\right) \cdot \frac{(Var(V_j))^{\frac{1}{2}}}{\sum_{i=1}^{n} (Var(V_i))^{\frac{1}{2}}}, \quad j = 1, \ldots, n. \tag{13}$$

The optimal solution (13) can be interpreted as follows: $M \cdot \mu_j$ is the expected space needed for the jth file (i.e. $E(\sum_{k=1}^{M} V_j^k)$). If we provide each file with the expected storage space it needs, there will be a space surplus of $S - M \cdot \sum_{i=1}^{n} \mu_i$. (If this "surplus" is negative, it reflects excess demand.) The jth file's share of the surplus is proportional to $(M \cdot Var(V_j))^{\frac{1}{2}}$, which is the standard deviation of the total space required by that file.

The optimal allocation of space can thus be performed in two stages:

Stage A. Assign each file its expected space demand.

Stage B. Allocate the remaining space according to the ratios of the standard deviations of demand. (If the remaining space is negative, delete from each file according to the ratios of standard deviations of demand.)

We have thus obtained a simple and convenient allocation rule. It requires a small number of parameters, and can be easily implemented. The optimal solution depends only on the expectations and standard deviations of the demand distributions.

In systems where optimization follows a reliability criterion, it will generally be true that $S > M \cdot \sum_{i=1}^{n} \mu_i$, because this criterion is usually employed when the costs of system failure are high. To avoid system failure, it has to be supplied with more space than the average demand. If $S < M \cdot \sum_{i=1}^{n} \mu_i$, then for every allocation vector $\mathbf{x} \in D$ there exists at least one i such that $x_i < E(\sum_{k=1}^{M} V_i^k)$. Under the above conditions, where M is large enough and $y_i \sim N(0, 1)$, this same i will satisfy $P\{\sum_{k=1}^{M} V_i^k \leq x_i\} < \frac{1}{2}$ and thus also $P\{N(\mathbf{x}) \geq M\} < \frac{1}{2}$. Clearly, such a system is not "reliable."

Let us now take a somewhat qualitative and intuitive look at the optimal allocation rule. If $S > M \cdot \sum_{i=1}^{n} \mu_i$,

then to obtain a reliable system we have to provide each file with at least the expected space it requires. This would exactly suffice if demand were deterministic. But since demand is stochastic, we would like to allocate excess space to each file to accommodate demand when it exceeds its expected value. The allocation of excess space should depend on the underlying uncertainty. The more uncertainty there is about a given file (i.e. the larger the standard deviation of demand at that file), the more space is required. It therefore seems reasonable to allocate the excess space according to the ratios of standard deviations.

The result for the case of $S < M \cdot \sum_{i=1}^{n} \mu_i$ is somewhat surprising because the larger the standard deviation of demand, the larger the portion of storage space that will be deleted from the allocated space. This has a logical explanation. We are actually "benefiting" from the existence of uncertainty. If demand in all files were deterministic, the system would fail with probability one because we do not have enough space to meet demand. In a stochastic environment, there is a positive probability that the system will not fail. Instances where demand exceeds its expected value are of no concern because expected space is not sufficient in any case. The larger the standard deviation of demand, the larger the probability that a given space will successfully meet demand. Thus, it seems reasonable to delete larger areas from files with larger standard deviations of demand.

5. Applications

5.1 Serial Files in an On-Line System

Consider an on-line system comprised of n serial files. File 1 is a journal file, in which all transactions are recorded for possible reconstruction or for statistical purposes. A transaction may add a record to each of the remaining $n - 1$ files. Let p_i denote the probability that a transaction updates file i ($i = 1, 2, \ldots, n$). We have

$$p_1 = 1, \quad 0 < p_i \leq 1, \quad i = 2, 3, \ldots, n.$$

We assume independence among the files. The record sizes for file i ($i = 1, 2. \ldots, n$) are iid random variables $\{Y_i^k, k = 1, 2, 3, \ldots\}$, where $Y_i^k \sim Y_i$ and $Var(Y_i) < \infty$. The problem is to allocate a given total space, S, among the files.

The optimal allocation depends on the first moments of the demand vector $\mathbf{V}$. In order to find these moments, we define the indicator random variables

$$A_i^k = \begin{cases} 1, & \text{transaction } k \text{ updates file } i, \\ 0, & \text{otherwise.} \end{cases}$$

For each $i = 1, 2, \ldots, n$, $\{A_i^k, k = 1, 2, 3, \ldots\}$ are iid., and $A_i^k \sim A_i$ where

$$P\{A_i = 1\} = p_i = 1 - P\{A_i = 0\}.$$

The demand generated by the kth transaction in file i is equal to $A_i^k \cdot Y_i^k$, so

$$\mathbf{V} = (A_1 Y_1, A_2 Y_2, \ldots, A_n Y_n), \tag{14}$$

where A_i and Y_i are independent for all $i = 1, 2, \ldots, n$. It follows that

$$\mu_i = E(V_i) = E(A_i)E(Y_i) = p_i E(Y_i), \tag{15}$$

$$E(V_i^2) = E(A_i Y_i^2) = p_i E(Y_i^2), \tag{16}$$

hence

$$Var(V_i) = E(V_i^2) - \mu_i^2 = p_i^2\, Var(Y_i) + p_i(1 - p_i)E(Y_i^2). \tag{17}$$

The first term of (17) is the contribution of the variation in the record size, Y_i, to the variance. The second term is due to the uncertainty in the *selection* of file i to be updated. It is nullified when $p_i = 1$.

An alternative derivation of (15) and (17) may be obtained by using the Laplace-Stieltjes transforms

$$\tilde{Y}_i(s) = E(e^{-sY_i}), \quad \tilde{V}_i(s) = E(e^{-sV_i}), \tag{18}$$

defined for $s \geq 0$. We present it as an introduction to the use of more complicated transforms in the second example.

LEMMA 4. *For* $s \geq 0$,

$$\tilde{V}_i(s) = p_i \tilde{Y}_i(s) + (1 - p_i), \quad i = 1, 2, \ldots, n. \tag{19}$$

PROOF. By conditioning on A_i^k we obtain

$$\begin{aligned} \tilde{V}_i(s) &= E(e^{-s \cdot V_i^k}) \\ &= P\{A_i^k = 1\} \cdot E(e^{-s \cdot V_i^k} | A_i^k = 1) + P\{A_i^k = 0\} \cdot 1 \\ &= p_i \cdot \tilde{Y}_i(s) + (1 - p_i). \end{aligned}$$

The moments of V_i as given by (15) and (16) may now be easily obtained by using the fact that

$$E(V_i) = -\tilde{V}_i'(0), \quad E(V_i^2) = V_i''(0). \qquad \square$$

The optimal allocation may now be found by a straightforward application of the results of sections 3 and 4. We solve the following numerical example: a total space of $S = 24$ megabytes is available for allocation among $n = 3$ files. Each transaction, which is 80 bytes long, is written on the journal file. Hence, $p_1 = 1$ and Y_1 is equal to 80 bytes with probability 1. A record is added to file 2 with probability $p_2 = \frac{1}{2}$. There are two record types, with lengths of 80 and 120 bytes, and Y_2 equals 80 or 120 bytes with equal probabilities. A record is added to file 3 with proability $p_3 = \frac{1}{4}$. These record lengths are assumed to have an exponential distribution with mean $1/\lambda = 100$ bytes. We summarize the relevant data and computations in Table I.

The expected values $E(Y_i)$ and variances $Var(Y_i)$ are

Table I. Data and computations for numerical example.

file i	P_i	$E(Y_i)$	$Var(Y_i)$	$\mu_i = E(V_i)$	$Var(V_i)$
1	1	80	0	80	0
2	$\frac{1}{2}$	100	400	50	2700
3	$\frac{1}{4}$	100	10000	25	4375

easily obtained for the given distributions of Y_i. The moments of V_i, μ_i and $Var(V_i)$, are computed from eqs. (15) and (17), respectively.

In order to maximize the *expected lifetime of the system* or the expected number of transactions processed until reorganization, the available storage space has to be allocated according to the proportion

$$\mu_1 : \mu_2 : \mu_3 = 80 : 50 : 25,$$

so

$$x_1^* = 24.80/155 = 12.39 \text{ megabytes},$$

$$x_2^* = 7.74 \text{ megabytes},$$

$$x_3^* = 3.87 \text{ megabytes}.$$

Thus, the journal file captures most of the available space, while the allotment of file 3 is less than 4 megabytes.

Now, consider the *reliability criterion* (which is usually suitable for on-line systems), and suppose the system is designed to process $M = 100{,}000$ transactions. Following the allocation rule developed in Section 4, we first allocate to file i the expected demand $M \cdot \mu_i$ $(i = 1, 2, 3)$. This yields 8 megabytes for the journal file, 5 megabytes for file 2, and 2.5 megabytes for file 3. The remaining 8.5 megabytes are allocated according to the proportions of standard deviations, which are 0: 0.785: 1. Hence, we obtain

$$x_1^* = 8 \text{ megabytes},$$

$$x_2^* = 8.7 \text{ megabytes},$$

$$x_3^* = 7.3 \text{ megabytes}.$$

Due to the allowance for uncertainty, the share of file 3 has increased considerably in comparison with its allowance under the proportionate allocation rule. This was done at the expense of the journal file whose demand is deterministic.

5.2 Allocation of PDS Libraries

A partitioned dataset (PDS) consists of several sequential subfiles called *members* [4]. Each member is made up of one or more records. The dataset also includes a *directory* containing the name, address and other features of each member. The directory enables direct access to the members of the PDS. The records of members are stored sequentially, so the PDS is a serial file.

A new member is created by writing the data sequentially, following the current end of the PDS, and storing a directory entry containing its name and address. Deletion of a member is performed by removing its name from the directory; however, the space used by the deleted member cannot be reused until the dataset is reorganized. When a member is changed, all its records are rewritten following the end of the file, and its directory entry is updated to point to its new location; the space used in its old location is not released.

The partitioned organization is usually used for program libraries. The program library is organized as a PDS whose members are separate programs or subprograms. A user usually maintains two program libraries: a source library, containing the source cards of the programs; and a library of load modules, from which the programs are loaded for execution.

When a program is to be changed or added, the source library is updated first. The updated member recorded on the source library is then compiled. If the compilation ends successfully, a member is created and stored in the library of load modules. In case of unsuccessful compilation, no space is demanded in the load-module library. Eventually, one of the libraries runs out of space and has to be reorganized. It is customary to reorganize both libraries together, since the setup cost of a reorganization is high.

Suppose that the libraries are to be allocated, and the available storage space is S. We wish to allocate S between the two libraries so that the expected time until reorganization is maximized. For that purpose, we have to compute the vector μ of expected demands per job.

Let V_1^k be the area needed in the source library for the kth job. We assume that $\{V_1^k, k = 1, 2, 3, \ldots\}$ are iid, and $V_1^k \sim V_1$. Let p be the probability of a successful compilation. With probability $1 - p$, compilation is unsuccessful, and the space needed in the library of load modules is $V_2^k = 0$. With probability p, a load module is created and $V_2^k > 0$.

It would be unreasonable to assume that V_2^k is independent of V_1^k, since it is likely that a longer source program creates a larger load module. We assume that if compilation is successful,

$$V_2^k = B^k V_1^k,$$

where $\{B^k, k = 1, 2, 3, \ldots\}$ are iid random variables and $B^k \sim B$. The "compression factor" B represents the ratio of program size in machine code to the total length of the source code. Its distribution depends on the programming language and on the programming habits. We further assume the B^k and V_1^k are independent.

The distribution of $\mathbf{V}$ is characterized by its two-dimensional Laplace-Stieltjes transform, defined by

$$\tilde{V}(\mathbf{s}) = \tilde{V}(s_1, s_2) = E[e^{-\mathbf{s}\cdot\mathbf{V}}] = E[e^{-s_1V_1 - s_2V_2}].$$

For the system described, we prove

LEMMA 5. *The Laplace-Stieltjes transform* $\tilde{V}(\mathbf{s})$ *is given by*

$$\tilde{V}(\mathbf{s}) = pE\tilde{V}_1(s_1 + Bs_2) + (1 - p)\tilde{V}_1(s_1) \tag{20}$$

where

$$\tilde{V}_1(s_1) = E[e^{-s_1V_1}]$$

is the (one-dimensional) Laplace-Stieltjes transform of V_1.

PROOF. By conditioning on the success of compilation, we obtain

$$\tilde{V}(\mathbf{s}) = pE[e^{-\mathbf{s}\cdot\mathbf{V}}|^{\text{successful}}_{\text{compilation}}] + (1-p)\cdot E[e^{-\mathbf{s}\cdot\mathbf{V}}|^{\text{unsuccessful}}_{\text{compilation}}]. \quad (21)$$

In case of a successful compilation,

$$\mathbf{V} \sim (V_1, BV_1),$$

hence

$$E[e^{-\mathbf{s}\cdot\mathbf{V}}|^{\text{successful}}_{\text{compilation}}] = E(e^{-s_1\cdot V_1 - s_2\cdot BV_1}) = E\tilde{V}_1(s_1 + Bs_2). \quad (22)$$

If compilation ends unsuccessfully,

$$\mathbf{V} \sim (V_1, 0),$$

$$E[e^{-\mathbf{s}\cdot\mathbf{V}}|^{\text{unsuccessful}}_{\text{compilation}}] = E[e^{-s_1 V_1}] = \tilde{V}_1(s_1). \quad (23)$$

Equation (20) follows by substituting (22) and (23) into (21). □

COROLLARY 1. *The expected demand vector is given by*

$$\mu = -\Delta\tilde{V}(\mathbf{s})|_{\mathbf{s}=0} = (E(V_1), \quad pE(B)E(V_1)) \quad (24)$$

It follows that in the optimal allocation

$$x_1^*:x_2^* = \mu_1:\mu_2 = 1:pE(B),$$

so only p and $E(B)$ have to be estimated.

Received August 1978

References

1. Maruyama, K., and Smith, S.E. Optimal reorganization of distributed space disk files. *Comm. ACM 19*, 11 (Nov. 1976), 634–642.
2. Mendelson, H. Optimal utilization of stochastically growing data-Bases. M.Sc. Th., Tel Aviv U., 1977. (In Hebrew.)
3. Mendelson, H., Pliskin, J.S., and Yechiali, U. A stochastic allocation problem. (To appear in *Operations Research.*)
4. OS Data Management Services Guide. IBM Form No. GC26-3746-2, IBM Systs. Ref. Libraries, White Plains, N.Y., 1973.
5. Ross, S.M. *Applied Probability Models With Optimization Applications.* Holden-Day, San Francisco, 1970.
6. Schneiderman, B. Optimum data-base reorganization points. *Comm. ACM 16*, 6 (June 1973), 362–365.
7. Van Der Pool, J.A. Optimum storage allocation for initial loading of a file. *IBM J. Res. Develop. 16* (1972), 579–586.
8. Van Der Pool, J.A. Optimum storage allocation for a file in steady state. *IBM J. Res. Develop. 17* (1973), 27–38.