SOME DISCRETE OPTIMIZATION PROBLEMS WITH SECONDARY STORAGE APPLICATIONS*

by

C. K. Wong

October, 1976 TR - 62

*This is an outline of a series of lectures delivered in the Department of Computer Sciences at the University of Texas at Austin in September 1976.

DEPARTMENT OF COMPUTER SCIENCES
THE UNIVERSITY OF TEXAS AT AUSTIN

I. PLACEMENT PROBLEMS

Given a set of objects $R_1, R_2, \ldots, R_n$ with associated positive numbers $p_1, p_2, \ldots, p_n$, $\sum_{i=1}^{n} p_i = 1$, the problem is to place these objects at the points $0, \pm 1, \pm 2, \ldots$ on the real line in such a way that

$$D = \sum_{i,j} p_i p_j \, d(i,j) \tag{1}$$

is minimized, where $d(i,j)$ is the Euclidean distance between objects i and j.

This corresponds to the problem of positioning records in a storage medium to minimize the expected access time when the medium can be modelled as linear. (p_i corresponds to the relative access frequency of record R_i.) For example, in the case of a disk when minimization of seek time is of interest, it is useful to view the cylinders as forming a linear store [7,12,22].

This problem was solved in a classical mathematical work by Hardy-Littlewood-Pólya [8]. The solution is simply to palce the object with the largest p and then repetitively to place the object with the next largest p alternating between the position immediately to the left (right) of those already placed and the position immediately to the right (left) [2, 7,8]. This will be referred to as the alternating algorithm or placement.

An immediate generalization of this problem is to consider a plane with points (i,j) where $i,j = 0, \pm 1, \pm 2, \ldots$, and $d(i,j)$ corresponds to different distance functions defined on the plane [10]. For example, let $\xi = (x_1, y_1)$ $\eta = (x_2, y_2)$ be two points in the plane, define

$$m_\alpha(\xi,\eta) = (|x_1-x_2|^\alpha + |y_1-y_2|^\alpha)^{1/\alpha}, \quad 1 \leq \alpha \leq \infty. \tag{2}$$

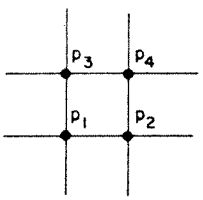
By $\alpha=\infty$ we mean

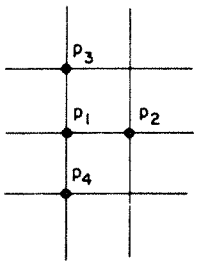
$$m_\infty(\xi,\eta) = \max(|x_1-x_2|, |y_1-y_2|). \tag{3}$$

Then we can define $d(i,j) = m_\alpha(\xi,\eta)$, if objects i,j are located at points ξ,η. α is a parameter depending on the application. For example, $\alpha=\infty$ corresponds to the case of a recent archive storage where accessing is done by an X-Y mechanism [23].

We consider the case $\alpha=2$ first. This corresponds to the Euclidean distance function.

We note that the optimal solution depends on the values of p rather than on their rankings alone as in the 1-dimensional case. The following example illustrates this fact.





$(p_1,p_2,p_3,p_4) = (0.33,0.32,0.31,0.04)$ $(p_1,p_2,p_3,p_4) = (0.70,0.15,0.10,0.05)$

Figure 1

These are optimal placements for the respective p-vectors.

We now propose a simple ranking-dependent heuristic which is asymptotically optimal (as n tends to infinity). This heuristic is a generalization of the alternating placement in the 1-dimensional case, namely, placing the object with largest p and then filling "shells" of points which are equidistant from the center with a set of objects whose p's are the next largest. This heuristic will be referred to as "shell" algorithm.

To prove this is asymptotically optimal, we assume $p_1 \geq p_2 \geq, \ldots, \geq p_n$ and define

$$\Delta_i = p_i - p_{i+1}, \quad 1 \leq i < n$$

and

$$\Delta_n = p_n,$$

so that

$$p_i = \sum_{r=i}^{n} \Delta_r \quad \text{and} \quad \sum_{r=1}^{n} r\Delta_r = \sum_{i=1}^{n} p_i = 1.$$

D can be rewritten as

$$D = \sum_{i=1}^{n} \sum_{r=i}^{n} \Delta_r \left(\sum_{j=1}^{n} \sum_{s=j}^{n} \Delta_s \right) d(i,j)$$

$$= \sum_{r=1}^{n} \sum_{s=1}^{n} \Delta_r \Delta_s E_{rs},$$

where

$$E_{rs} = \sum_{i=1}^{r} \sum_{j=1}^{s} d(i,j).$$

Thus, the effect of the placement algorithm is localized to a term, E_{rs}, which is independent of p.

Let $D(opt)$, $E_{rs}(opt)$, $D(shell)$, $E_{rs}(shell)$ be the values produced by the optimal and the "shell" algorithms respectively. If one can show that

$$E_{rs}(shell) \leq E_{rs}(opt) + crs, \tag{4}$$

then

$$D(shell) = \sum_{r=1}^{n} \sum_{s=1}^{n} \Delta_r \Delta_s E_{rs}(shell)$$

$$\leq \sum_{r=1}^{n} \sum_{s=1}^{n} \Delta_r \Delta_s E_{rs}(opt) + c \sum_{r=1}^{n} \sum_{s=1}^{n} rs \, \Delta_r \Delta_s$$

$$= D(opt) + c.$$

To prove (4), we consider the continuous analogue of E_{rs} for which it is relatively easy to find the optimal solution. Recall that if we look at the first r and s $(r \leq s)$ points in the configuration resulting from the "shell" algorithm, then $E_{rs}(shell) = \sum_{i=1}^{r} \sum_{j=1}^{s} d(i,j)$. If we now replace each point by a unit square with center at this point, then we have two regions ω_0, ω_1 with areas r,s, respectively, and $\omega_0 \subset \omega_1$. Let

$$E_{r,s}^{cont}(\text{shell}) = \iint_{x\in\omega_1} \iint_{y\in\omega_0} d(x,y).$$

Let $E_{r,s}^{cont}(\text{opt})$ be defined in the same way, and let

$$E_{rs}^{cont} = \min_{\substack{\omega_0\subset\omega_1 \\ \text{area } \omega_0=r \\ \text{area } \omega_1=s}} \int_{x\in\omega_1} \int_{y\in\omega_0} d(x,y). \tag{5}$$

If one can show that

$$E_{rs}(\text{shell}) \leq E_{rs}^{cont}(\text{shell}) + \beta\ rs, \tag{6}$$

$$E_{rs}^{cont}(\text{shell}) \leq E_{rs}^{cont} + \gamma\ rs, \tag{7}$$

$$E_{rs}^{cont} \leq E_{rs}^{cont}(\text{opt}) \leq E_{rs}(\text{opt}) + \delta\ rs, \tag{8}$$

(4) will follow immediately. (6), (8) are easy. To show (7), one has to solve the problem (5). Fortunately, by geometric arguments, one can show that the optimal solution is obtained when ω_0, ω_1 are concentric disks. Based on this, (7) can be proved [10].

However, when $\alpha \neq 2$, the argument does not go through.

In fact, we shall demonstrate the nonexistence of asymptotically optimal ranking-dependent heuristics in general except in the uniform case when $p_i = \frac{1}{n}$ for all i. In this case, $\Delta_n = \frac{1}{n}$ and

$\Delta_i = 0$, $1 \leq i < n$. We have only to consider E_{nn} and the continuous optimization problem

$$\min_{\substack{\omega \\ \text{area } \omega = n}} \int_{\xi,\eta \in \omega} m_\alpha(\xi,\eta). \tag{9}$$

A necessary condition for an otpimal solution to (9) can be stated as follows: Let ω be an optimal region and a be a point on the boundary of ω. Then $\int_{\eta \in \omega} m_\alpha(a,\eta)$ is independent of a [10]. This condition gives a differentio-integral equation of the boundary curve of the optimal region. For the cases of interest: $\alpha=1$ and $\alpha=\infty$, let $v=f(u)$ be the equation of the boudnary curve of the optimal region in the first quadrant, then the necessary condition reduces to the following differentio-integral equations:

$$f'(u)\ [\frac{n}{4} - \int_0^u f(x)\,dx + uf(u)\,] + \int_0^u f(x)\,dx = 0, \quad \text{for} \quad \alpha=1. \tag{10}$$

$$[n + 2(f^2(u)-u^2)]\ f'(u) + 8\int_0^u f(x)\,dx - 4uf(u) = 0, \quad \text{for} \quad \alpha=\infty. \tag{11}$$

These equations are solved numerically in [10]. They turn out to be different from the unit circles in the respective distance functions, as intuition might have suggested. The whole boundary may then be obtained by reflecting the curve in the first quadrant with respect to the x- and y- axes.

For the case of arbitrary p, let us look at the following special family of p-vectors:

$$p_1 = z, \quad p_i = \frac{1-z}{n-1}, \quad i = 2,\ldots,n.$$

Consider the continuous optimization problem

$$\min_{\substack{\omega_0 \subset \omega_2 \\ \text{area } \omega_0 = 1 \\ \text{area } \omega_2 = n}} \iint_{\xi,\eta\in\omega_2} p(\xi)p(\eta)\, m_\alpha(\xi,\eta) \tag{12}$$

where

$$p(x) = \begin{cases} z, & x\in\omega_0, \\ \\ \frac{1-z}{n-1}, & x\in\omega_2-\omega_0. \end{cases}$$

Then the difference between the optimal values of (1) and (12) is bounded by a constant. We therefore have to consider (12) only. To compute $I = \iint_{\xi,\eta\in\omega_2} p(\xi)p(\eta)\, m_\alpha(\xi,\eta)$, we introduce the coordinate transformation

$$\xi = (x,y) \mapsto \xi' = (x',y') = (\frac{1}{\sqrt{n}}\, x, \quad \frac{1}{\sqrt{n}}\, y).$$

Then the area will be shrunk by a factor of $\frac{1}{n}$ and distance by a factor $\frac{1}{\sqrt{n}}$. Define

$$q(\xi') = n\, p(\xi) \quad \text{and} \quad I_1 = \int_{\xi',\eta' \in \omega_2'} \int q(\xi')q(\eta')\, m_\alpha(\xi',\eta').$$

Then

$$I = \sqrt{n}\, I_1. \tag{13}$$

Now ω_0' has area $\frac{1}{n}$ and ω_2' has area 1. One can show that

$$I_1 = 2\, z(1-z)A + (1-z)^2 B + O(\frac{1}{\sqrt{n}}),$$

where A is the average distance between points in ω_0' and $\omega_2'-\omega_0'$ and B is that between points in $\omega_2'-\omega_0'$. Thus,

$$I = \sqrt{n}\, [2\, z(1-z)A + (1-z)^2 B] + O(1).$$

For $\alpha=1$, if z is very close to 1, the optimal solution to (12) is a region which minimizes A. Hence, the optimal ω_0 is a unit disk in this distance function. On the other hand, if z is $\frac{1}{n}$, we have the uniform case again, where the optimal solution to (12) is a region which minimizes B and is not a unit disk. For these two cases, I differs by a quantity proportional to $\sqrt{n}$. If there is a ranking-dependent heuristic, for our family of p-vectors, the optimal region remains the same for all values of z. Clearly, it cannot be asymptotically optimal.

Similar argument for $\alpha = \infty$. In the case of $\alpha=2$, the unit disk minimizes A, B simultaneously.

Open Problems

(1) Instead of an infinite plane, if we have an $m \times m$ square and m^2 objects, how to place them so that (1) is minimized? Some experiments with variations of the "shell" algorithm were performed [23]. They seemed to be very good. But no analytic results were obtained.

(2) Instead of independent access probabilities $p_1,\ldots,p_n$, if we assume that they form a first order Markov chain, what will be the optimal placement? Let p_{ij} denote the conditional probability to access record j if record i has just been accessed and $q_1,\ldots,q_n$ the stationary probability. Then the counterpart of (1) will be $D = \sum_{i,j} q_i\, p_{ij}\, d(i,j)$. Intuitively, one may want to arrange the records alternately according to the ranking of $q_1,\ldots,q_n$. However, it can easily be shown that it is not always optimal. Then what is the performance of such a heuristic?

II. PARTITION AND PLACEMENT PROBLEMS

Another generalization of the original 1-dimensional placement problem is the following partition problem [22]:

Given a set of objects $R_1,R_2,\ldots,R_{mk}$ with associated positive numbers $p_1,p_2,\ldots,p_{mk}$, $\sum_{i=1}^{mk} p_i = 1$, we want to first partition them into m group of k objects each. Each group is associated with a number q_j which is the sum of the k numbers in the group. Then we want to place the groups such that

$$D = \sum_{i,j}^{m} q_i\, q_j\, d(i,j) \qquad (1')$$

is minimized. Clearly, only the partition problem needs consideration since the placement problem has been solved in Section I.

This problem corresponds to the situation of allocating records to a disk, which consists of m cylinders, each with capacity of k records. The solution to the problem is simple: Assume $p_1 \geq p_2 \geq \ldots \geq p_{mk}$. Then group the first k numbers in one group, the second k numbers in another group and so on. We then obtain a q-vector $\underline{q}_0 = (q_1,\ldots,q_m)$ such that $q_1 \geq \ldots \geq q_m$. To show this is optimal, we need the concept of vector majorization and a powerful theorem by Schur [11,15,17,20,22].

Definition. If $\underline{v} = (v_1,\ldots,v_m)$ and $\underline{u} = (u_1,\ldots,u_m)$ are such that

$v_1 \geq \ldots \geq v_m$, $u_1 \geq \ldots \geq u_m$ and $\sum_{i=1}^{k} v_i \geq \sum_{i=1}^{k} u_i$, $k = 1,2,\ldots,m-1$,

$\sum_{i=1}^{m} v_i = \sum_{i=1}^{m} u_i$, $\underline{v}$ is said to majorize $\underline{u}$, in symbol, $\underline{v} \succ \underline{u}$.

Definition. A real-valued function of m real variables $\phi = \phi(x_1,\ldots,x_m)$ is said to be a Schur function if

$$(x_i - x_j)\left(\frac{\partial \phi}{\partial x_i} - \frac{\partial \phi}{\partial x_j}\right) \geq 0 \quad \text{for all } i,j. \tag{14}$$

Theorem. (Schur) If $\phi(x_1,\ldots,x_m)$ is a Schur function and if $\underline{v} \succ \underline{u}$, then $\phi(v_1,\ldots,v_m) \geq \phi(u_1,\ldots,u_m)$.

Note that given $p_1 \geq p_2 \geq \ldots \geq p_{mk}$, among all possible partitions into m groups, the vector $\underline{q}_0$ obtained by our algorithm majorizes all other resulting vectors. It remains to show that $-D$ is a Schur function of

$q_1,\ldots,q_m$. To demonstrate the approach, we shall assume $m = 5$, and q_4,q_2,q_1,q_3,q_5 is the placement. Then

$$D = 2(q_1q_2 + q_1q_3 + 2q_1q_4 + 2q_1q_5 + 2q_2q_3 + q_2q_4$$

$$+ 3q_2q_5 + 3q_3q_4 + q_3q_5 + 4q_4q_5).$$

Let $i = 1$, $j = 2$, then

$$\frac{\partial D}{\partial q_2} - \frac{\partial D}{\partial q_1} = 2(q_1 + 2q_3 + q_4 + 3q_5 - q_2 - q_3 - 2q_4 - 2q_5) \geq 0,$$

since $q_1 \geq q_2$ and $q_3 \geq q_4$. This can be shown to be true for all i,j. (14) is thus satisfied and $-D$ is a Schur function. Hence, the original optimization problem reduces to the verification of a given function being Schur.

This powerful technique can also be applied in other cases. We shall next discuss the case of improving program behavior in a paging environment by improving locality. Given mk records $R_1,R_2,\ldots,R_{mk}$ with request probabilities $p_1,p_2,\ldots,p_{mk}$, $\sum_{i=1}^{mk} p_i = 1$, which are to reside in m pages, each with capacity k records, the problem is to allocate them such that the subsequent processing of paging requests will be optimal. We shall use two optimality criteria. Let the set of pages be enumerated by $\{1,2,\ldots,m\}$. Let $\{r_t\}$, $t = 0,\pm1,\pm2,\ldots$ be the sequence of page requests, i.e., r_t is the page referenced at time t and it takes value in the set $\{1,2,\ldots,m\}$.

The first criterion is defined as follows. Let N be a fixed positive integer, and let $K_t^{(N)}$ be the number of distinct values of the set $\{r_{t-N+1},\ldots,r_{t-1},r_t\}$, i.e., the cardinality of the set. Then $K_t^{(N)}$ is a random variable. Since its distribution is independent of t, we write it as $K^{(N)}$. Its expectation $\mathcal{E}K^{(N)}$ will be used as the objective function to be minimized.

The second criterion is defined as follows. Suppose the request at time t is i, and r_τ is the preceding request for the same page i. Then the number of distinct values of the set $\{r_{\tau+1},\ldots,r_{t-1}\}$, denoted K, is defined as the distance at time t, designated P_t. If this set is empty, K is defined as zero. P_t is a random variable with a distribution independent of t. It will be written as P, and its expectation $\mathcal{E}P$ will be the second objective function to be minimized.

The first criterion corresponds to the notion of working set size; the second corresponds to that of LRU distance. Both are indicators of the number of distinct pages referenced locally, where "locally" in the latter case is defined by the interval between successive references to the same page, while in the former case the interval is fixed but has arbitrary length. By minimizing $\mathcal{E}P$ and $\mathcal{E}K^{(N)}$, we are minimizing the amount of pages which must reside in main memory in order to continue program execution efficiently.

Suppose $p_1 \geq p_2 \geq \ldots \geq p_{mk}$, as before, if we group the first k records in one page, then the second k records in another page and so on, with resulting page request probabilities $q_1 \geq q_2 \geq \ldots \geq q_m$, then it turns out both

$\mathcal{E}K^{(N)}$ and $\mathcal{E}P$ are minimized. The next step is to show $-\mathcal{E}K^{(N)}$ and $-\mathcal{E}P$ are Schur functions which can easily be done [22].

As a footnote, it should be pointed out that not only $-\mathcal{E}K^{(N)}$ is a Schur function, but also the distribution is a Schur function. In fact,

$$F_j = P_r[K^{(N)} \leq j]$$

$$= \sum_{i=1}^{j} (-1)^{j-i} \binom{m-i-1}{j-i} S_i,$$

where S_i, $i = 1,2,\ldots,m$, are the following symmetric functions:

$$S_1 = x_1^N + \ldots + x_m^N$$

$$S_2 = (x_1+x_2)^N + (x_1+x_3)^N + \ldots + (x_{m-1}+x_m)^N$$

$$\vdots$$

$$S_m = (x_1+\ldots+x_m)^N,$$

and $-F_j$ is a Schur function [20].

A family of preservation theorems in mathematical statistics can be proved in exactly the same fashion [13,14,16].

So far, the algorithms we proposed are either optimal or asymptotically optimal for all input. Next, we shall discuss a dynamic placement problem and propose a heuristic which has asymptotically optimal average value.

As discussed before, given a set of objects and their associated numbers, the alternating placement will minimize (1). We now consider the

problem that not all objects (and their numbers) are known before hand, but rather they come in one by one and they have to be placed as soon as they come in. The objective is still the minimization of (1). This problem is motivated by the space allocation problem for the minidisks of users of VM/370 [12]. As each user logs on, space for his minidisk must be obtained on the on-line disks. For each user it is possible to estimate his activity based on accounting information from past usage of the system. But since it is in general not possible to know exactly who will log on at a given time, relative frequency of access is not known prior to the allocation of individual disks. We assume that a sequence of users $\{u_t\}$, $1 \leq t \leq n$, arrive at the system at distinct points in time. Associated with each user, u_t, is a frequency of use, f_t. The f_t's are assumed to be independent, identically distributed random variables. We further assume that f_t is uniformly distributed on the interval (0,1]. Each user is allocated a vacant space, i, in a linear store where locations have been numbered as follows:

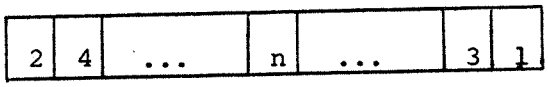


Figure 2

After complete allocation of all users, user u_t will have relative frequency $f_t \Big/ \sum_{t=1}^{n} f_t$ and the expected access time is $D = \sum_{ij} \frac{f_i f_j}{(\Sigma f_i)^2} d(i,j)$, where f_i now corresponds to the user at location i.

We propose the following dynamic allocation algorithm: Partition the interval $(0,1]$ into n subintervals: $(0, \frac{1}{n}], (\frac{1}{n}, \frac{2}{n}], \ldots, (\frac{n-1}{n}, 1]$. If the frequency f_1 of the first user u_1 belongs to the j-th subinterval, allocate location j to him. We now partition the interval $(0,1]$ into $(n-1)$ subintervals: $(0, \frac{1}{n-1}], (\frac{1}{n-1}, \frac{2}{n-1}], \ldots, (\frac{n-2}{n-1}, 1]$. If f_2 of the second user belongs to the k-th subinterval, allocate the k-th smallest labelled location among the remaining $(n-1)$ locations to this user, and so on. Let D_h donote the expected access time by this algorithm and D_o that by the alternating placement if all frequencies are known before hand. Of course, $D_o \leq D_h$. But surprisingly, the average values $\overline{D_o}$, $\overline{D_h}$ over all possible samples $\{f_t\}$, $1 \leq t \leq n$, from the distribution are asymptotically the same (as n tends to infinity). Thus, for n large, the dynamic allocation algorithm is nearly optimal. In fact, computation shows that

$$\overline{D_o} = \frac{7}{30} n + O(1)$$

and

$$\overline{D_h} = \frac{7}{30} n + a \ln n + O(1),$$

where a is a small constant.

For comparison, if we do not use any allocation algorithm at all, but rather assign space to the users randomly, then the average value of the expected access time will be

$$\overline{D_r} = \frac{n}{3} + O(1).$$

Thus, the dynamic allocation algorithm indeed brings forth some improvement.

Open Problems

(1) The last problem is a variation of the original 1-dimensional placement problem. We now consider another variation, namely, batch processing. Instead of processing the request one at a time, suppose we do it for a batch of b requests at a time. What then are the optimal placement of the records (with known access frequencies) and access algorithm to minimize the expected access time per request? Note that in the previous case when request is processed one at a time, no consideration of access algorithm is needed. In the present case, however, it is necessary to consider the way in which the b requests are processed. We therefore have a two stage problem. An intuitively appealing solution is as follows: Place the records by the alternating algorithm. Given b requests, find out the locations of the requested records, move to the nearest extreme location, and start processing the b requests in one pass. Let D be the expected access time per record, then $D \leq \frac{n}{b}$ is obvious. But more detailed analysis of the algorithm is not known.

(2) Given mk records with known access frequencies, how to allocate them in m linear stores, with capacity k each to minimize expected access time, assuming inter-storage movement takes no time? One obvious algorithm is as follows: Regard the storage system as a $m \times k$ matrix, fill it column by column in an alternating fashion. Clearly, this is not optimal.

For example, for $m=2$, $k=3$, the placement in Figure 3a has a smaller

0	.97	0
.01	.01	.01

Figure 3a

expected access time than that in Figure 3b, which is obtained from the

.01	.97	0
.01	.01	0

Figure 3b

alternating algorithm. However, what is the performance of this algorithm?

III. PACKING PROBLEMS

Given objects $R_1,\ldots,R_n$ with associated positive numbers $p_1,\ldots,p_n$, $\sum_{i=1}^{n} p_i = 1$ and sizes $y_1,\ldots,y_n$, $0<y_i<1$, we want to partition them into groups with total size less than or equal to 1 and if q_j is the sum of the access frequencies in group j, we seek to maximize $Q = \sum_j q_j^2$.

If $R_1,\ldots,R_n$ are data sets, $p_1,\ldots,p_n$ are their relative access frequencies and $y_1,\ldots,y_n$ are their sizes and the capacity of a page is 1, then q_j is the relative page access frequence for page j and

$C = \sum_j q_j(1-q_j) = 1 - \sum_j q_j^2$ is the expected number of page boundary crossings. Thus, the minimization of C is equivalent to the maximization of Q [19].

Again, optimal or even near-optimal solution to the problem is difficult to obtain. However, we shall present a simple heuristic with worst-case ratio of Q_o to Q_h equal to 2, where Q_o, Q_h are respectively the values of Q by the optimal algorithm and the heuristic.

Heuristic: Assume $\frac{p_1}{y_1} \geq \frac{p_2}{y_2} \geq \ldots \geq \frac{p_n}{y_n}$. Let $k = \max \{i \mid \sum_{j=1}^{i} y_j \leq 1\}$. Then assign $R_1,\ldots,R_k$ to the first page. (Break tie with random choice.) Repeat the process for the remaining records. Let

$$\tau = \underset{\substack{(p_1,\ldots,p_n)\\(y_1,\ldots,y_n)}}{\text{l.u.b.}} \frac{Q_o}{Q_h}.$$

Then it can be proved that $\tau=2$. By an application of Schur theorem, one can show that $\tau \leq 2$. To show $\tau \geq 2$, we construct the following example: Let $p_{2i-1} = y_{2i-1} = \frac{1}{2}$, for $1 \leq i \leq 2t$ and $p_{2i} = y_{2i} = \frac{1}{t}$, for $1 \leq i \leq 2t$. Then in the worst-case the heuristic may group p_{2i-1} and p_{2i} together in one page and $Q_h = 2t(\frac{1}{2} + \frac{1}{t})^2$. But clearly, $Q_o \geq t$. Thus $\frac{Q_o}{Q_h} \geq \frac{1}{2(\frac{1}{2} + \frac{1}{t})^2} \to 2$ as $t \to \infty$. If we now consider a more general objective function $Q^{(\alpha)} = (\sum_j q_j^{\alpha})^{1/\alpha}$, $2 \leq \alpha \leq \infty$, $(Q^{(\infty)} = \max(q_1,q_2,\ldots))$ and define $\tau^{(\alpha)}$ in the same way, then it can be shown that $\tau^{(\alpha)} = 2^{1-\frac{1}{\alpha}}$ for all α.

The above approach is typical of a worst-case analysis. Let us consider a somewhat dual problem in which Q is minimized instead of being maximized.

This time we want to allocate the n records to the m sectors of a drum to minimize expected latency time [3,4,5]. Let the relative access frequency to sector j be q_j after the allocation, then the objective function to be minimized is similar to (1):

$$L = \sum_{i,j} q_i \, q_j \, d(i,j), \tag{1"}$$

except that $d(i,j)$ now is the rotational distance given by

$$\begin{cases} d_{ij} = j-i-1 & 1 \le i < j \le m, \\ d_{ji} = m-j+i-1 & 1 \le i < j \le m, \\ d_{ii} = m-1. \end{cases}$$

L is the expected latency time to go from a sector to another. (1") can now be rewritten as

$$L = \frac{1}{2} \sum_{i,j} q_i \, q_j \, (d_{ij} + d_{ji})$$

$$= \frac{m-2}{2} + \frac{m}{2} \sum_{i=1}^{m} q_i^2.$$

Thus the minimization of L is equivalent to that of $Q = \sum_{i=1}^{m} q_i^2$.

Again, we have a very good heuristic: Assume the given records $R_1,\ldots,R_n$ have relative access frequencies $p_1,\ldots,p_n$ and $p_1 \geq \ldots \geq p_n$. Put them one by one into the m sectors. Each time the sector with the smallest current sum of p's is chosen. Break tie with random choice.

Let

$$\sigma = \underset{(p_1,\ldots,p_n)}{\text{l.u.b.}} \frac{Q_h}{Q_o}.$$

Then it can be shown that [3]

$$\frac{25}{24} \geq \sigma \geq \begin{cases} \frac{37}{36}, & m \text{ even} \\ \frac{83}{81}, & m = 3 \\ \frac{37}{36} - \frac{1}{36m}, & m \text{ odd and } m \geq 5, \end{cases}$$

i.e., $1.04 \geq \sigma \geq 1.03$ for large m. Thus the heuristic is at most 4% off optimum.

We can also generalize the objective function to $Q^{(\alpha)} = \left(\sum_{i=1}^{m} q_i^{\alpha} \right)^{1/\alpha}$ and define $\sigma^{(\alpha)}$ accordingly. For $\alpha=\infty$, this corresponds to the problem of sequencing independent tasks on multiple processors with execution times known considered by Graham [6]. He showed that $\sigma^{(\infty)} = \frac{4}{3} - \frac{1}{3m}$. We can show [3] that $\sigma^{(\alpha)} \to \sigma^{(\infty)}$ as $\alpha \to \infty$ and that

$$\sigma^{(\alpha)}(m) \leq \frac{3^\alpha - 2^\alpha}{\alpha} \left(\frac{\alpha - 1}{2.3^\alpha - 3.2^\alpha} \right)^{\frac{\alpha-1}{\alpha}}$$

$$\sigma^{\alpha}(m) \geq \max_{m'} \left(\frac{\left\lfloor \frac{m}{m'} \right\rfloor \left[(4m'-1)^\alpha + (m'-1)(3m'-1)^\alpha \right] + (m \bmod m')(3m')^\alpha}{m(3m')^\alpha} \right)^{1/\alpha},$$

where $m \bmod m' = m - m' \left\lfloor \frac{m}{m'} \right\rfloor$.

A much harder problem is when $n = mk$ and each sector has capacity k. A variation of the above heuristic has pretty good performance [5]. But complete analysis is not available yet.

So far, all the problems we have considered have objective functions in analytic form. Next we shall consider a family of graph problems and propose an asymptotically optimal heuristic [9].

Given a directed graph, let n be the number of nodes, e the number of edges and k the diameter of the graph, i.e., the number of edges in the longest path, we consider the following three problems:

(a) (n,k) problem: Given n,k, find a directed graph with n nodes, whose diameter is at most k such that the number of edges is minimized.

(b) (n,e) problem: Given n,e, find a directed graph with n nodes and at most e edges whose diameter is minimized.

(c) (e,k) problem: Given e,k, find a directed graph with at most e edges and diameter at most k such that the number of nodes is maximized.

The first problem comes from the following allocation problem: Given n pages of records, we want to assign page pointers to each page. These

pointers point to other pages. On the one hand, we want to minimize the total number of pointers since each pointer corresponds to a page fault. On the other hand, in order to guarantee the performance of the system as a whole, the following constraint is imposed: The number of pointers traced when going from one page to another should not be larger than k.

The second problem was first posed as an open problem in Berge's book [1]. The third one is a natural complement to the first two.

We shall solve the (n,k) problem first. Note that for $k = n-1$, a cycle with n nodes (and n edges) is the optimal solution. For $k=1$, a complete graph with n nodes is the optimal solution.

We propose the following heuristic construction: Let there be a loop having $\lceil k/2 \rceil + 1$ nodes. Pick one of the nodes and call it CENTER. As long as there are at least $\lfloor k/2 \rfloor$ nodes that have not yet been picked, pick $\lfloor k/2 \rfloor$ of them and, together with CENTER, add edges to make a loop. If there are less than $\lfloor k/2 \rfloor$ nodes unpicked, add edges so as to make a loop using those nodes and CENTER. Finally, assign the same direction to all edges in a loop. Such a graph will be called a flower graph. The following example is for $n = 15$, $k = 7$.

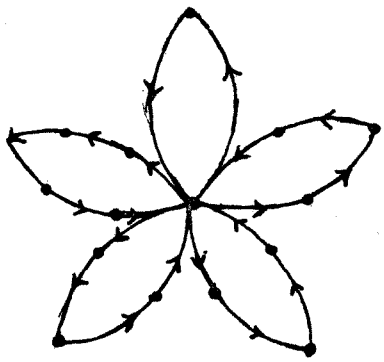


Figure 4

The number of edges is 19 in this case.

In general, the diameter of a flower graph is exactly k and the number of edges is $n + \left\lceil \frac{n-1 - \left\lceil \frac{k}{2} \right\rceil}{\left\lfloor \frac{k}{2} \right\rfloor} \right\rceil$.

If $e(n,k)$ denotes the minimum number of edges necessary to solve the (n,k) problem, then

$$e(n,k) \leq n + \left\lceil \frac{n-1 - \left\lceil \frac{k}{2} \right\rceil}{\left\lfloor \frac{k}{2} \right\rfloor} \right\rceil .$$

Using a careful counting argument, it can be shown that [9]

$$e(n,k) \geq n-1 + \left\lceil \frac{2(n-1)}{k} \right\rceil .$$

If k is even, the upper and lower bounds coincide. If k is odd, they are asymptotically the same as n tends to infinity. However, it is not always optimal. For example, for $n = 15$, $k = 7$ the following graph has 18 edges only:

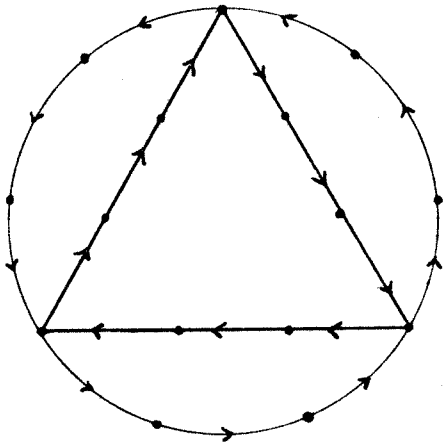
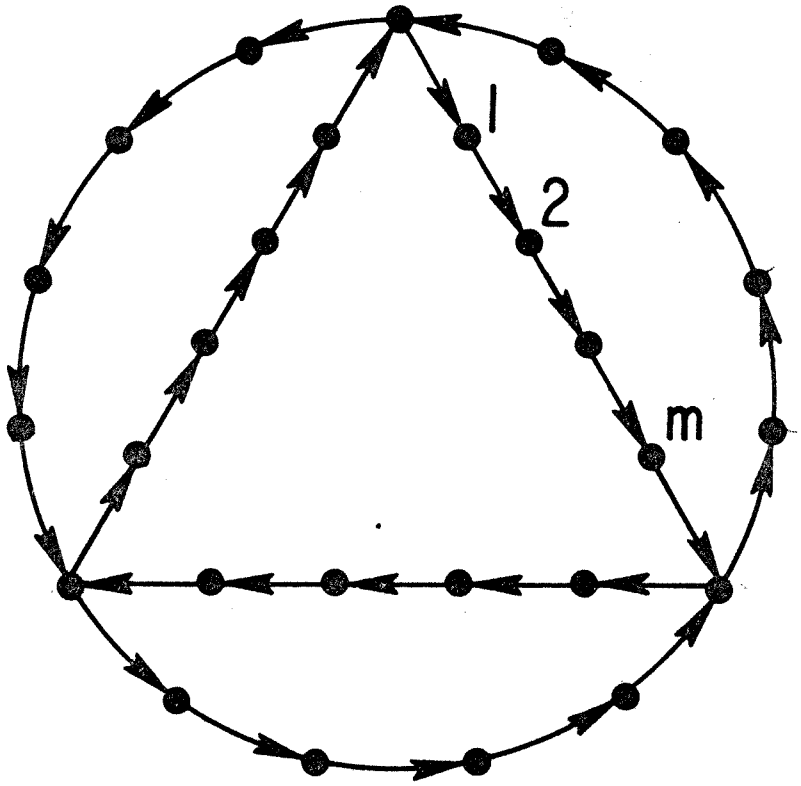


Figure 5

In general, a graph of the following form is called a circular graph:



A CIRCULAR GRAPH

Figure 6

For such a graph it can easily be seen that $n = 6m+3$, $e = 6m+6$, $k = 3m+1$. If k is even, a flower graph and a circular graph with the same n and k will have the same number of edges e. Furthermore, a circular graph is always optimal.

The same kind of construction works for the other two problems.

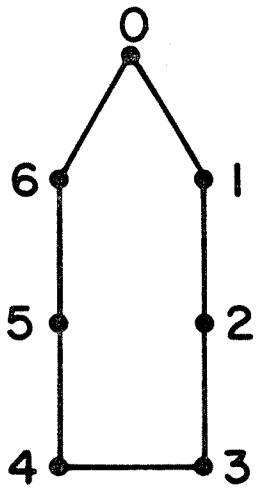
Open Problem

Given $p_1, p_2, \ldots, p_{mk}$, how to partition them into m groups of k each with sums $q_1, \ldots, q_m$ such that $\sum_{i=1}^{m} q_i^2$ is minimized? For $k=2$, an optimal solution is available: Assume $p_1 \geq p_2 \geq \ldots \geq p_{2m}$, then group p_i and p_{2m-i} together to form m groups. There are obvious generalizations of this algorithm but complete analysis is not known [5].

IV. DATA ARRANGEMENT IN MAGNETIC BUBBLE MEMORIES

We now turn to the problem of data arrangement and accessing in a new kind of secondary memory, namely, magnetic bubble memory. It has high density, fast access time (compared to disks), relatively low cost (compared to main memory) and is non-volatile [18,21].

It can be regarded as a long loop of cells as far as data accessing is concerned. Data items are stored in the cells. The only way to access an item is through an I/O port, i.e., the item has to be in a specific location before it can be accessed.

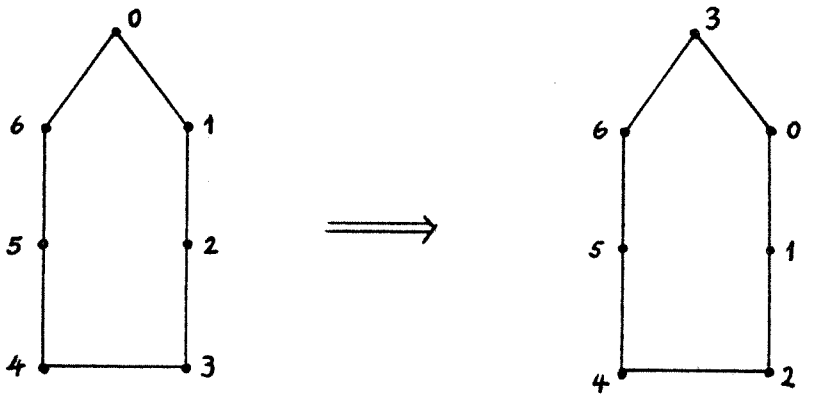


LOCATION LABELS OF A LOOP OF SIZE 7

Figure 7

We shall assume the I/O port has location zero. The items can be circulated along the loop, say, in a counter clockwise direction. Then if the size of the loop is n, the average access time will be $n/2$. To improve upon this, we notice that in general access frequencies of the items are not the same. To take advantage of this, one would like to keep the more frequently accessed items closer to the I/O port. Furthermore, one would like to achieve this automatically without prior knowledge of the access frequencies. One way to approximate this is to keep the latest accessed item at the I/O port and shift the other items accordingly. For example, suppose item i is located at location i for all i, and we want to access item k. Afterwards we would like to maintain the order of items in the following way: $k,0,1,\ldots,k-1, k+1,\ldots,n-1$ in a clockwise direction.

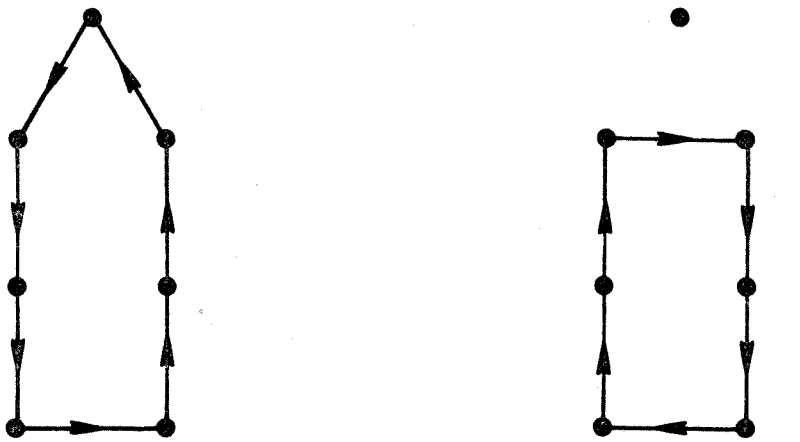
For example, when k=3, n=7:



BEFORE ACCESSING AFTER ACCESSING

Figure 8

In the long run, more frequently accessed items will tend to be closer to location 0. To accomplish such an ordering, we introduce a new operation to the momory. We therefore have the following two kinds of operations:



OPERATION a OPERATION e

Figure 9

Operation a means that in one unit of time, the item at each location is moved to the next location in a counter clockwise direction. Operation e means that in one unit of time the item at location 0 remains intact, but all other items are moved to their next locations in a clockwise direction as shown in the figure. To access item 3 and achieve the subsequent ordering as shown in Figure 8, we have only to apply the sequence of operations a^3e^3, i.e., 3 a's followed by 3 e's.

In general, to access an item at location k, we need k a's followed by k e's. However, this is not optimal (in the total number of operations) in general. For example, if n=7, and k=6, instead of 12 operations, one needs only 3, namely, eae. The optimal sequence is given as follows:

(1) For $1 \leq k \leq \left\lfloor \frac{6n-7}{8} \right\rfloor$, apply $a^k a^k$. (Total = 2k).

(2) For $\left\lfloor \frac{6n-7}{8} \right\rfloor \leq k \leq n-2$, apply $(eae)^{n-k-1}\, ea(aea)^{n-k-2}$.
(Total = 6(n-k)-7).

(3) For k=n-1, apply eae. (Total = 3).

The next question is how to realize a general permutation of items in memory. No optimal algorithm is known. But the following algorithm differs from optimum by only a constant factor. To illustrate its operation, we use a simple example. The following is the required permutation:

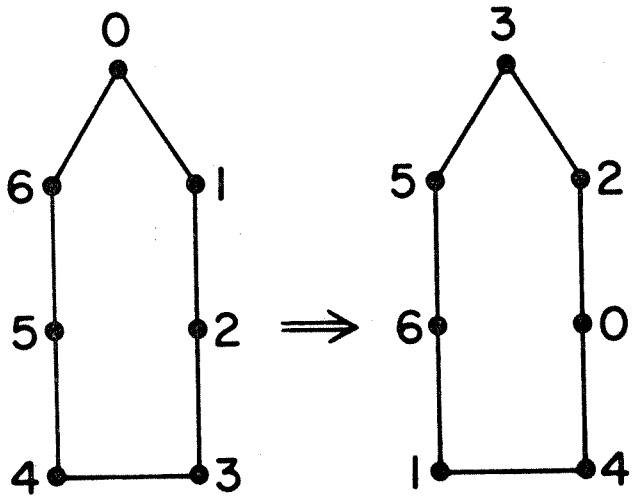


Figure 10

We shall use the previous accessing algorithm to bring item 3 to location 0 first. Then apply a, so that 3 is at location 6. (See Figure 11). Then repeat the same process by bringing item 2 to location 0, and so on.

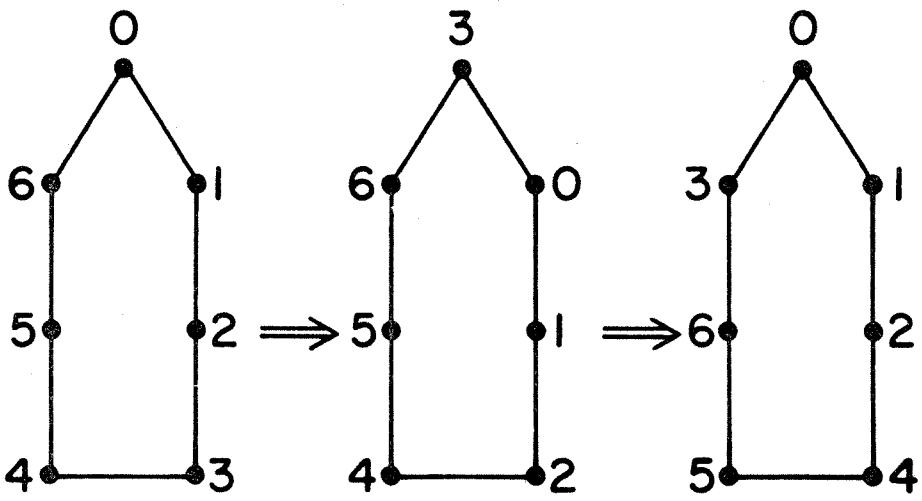


Figure 11

It can be shown [18] that for both the worst-case and the average case $O(n^2)$ operations are needed to realize an arbitrary permutation. We shall next show that $O(n^2)$ is also a lower bound for both cases. Thus, this algorithm differs from optimum by only a constant factor.

Let A be any algorithm which generates all permutations. Let $t_A(\pi)$ be the number of operations to generate π. Then

(a) There exists $\hat{\pi}$ such that $t_A(\hat{\pi}) \geq \frac{1}{8} n^2 - \frac{3}{8} n$.

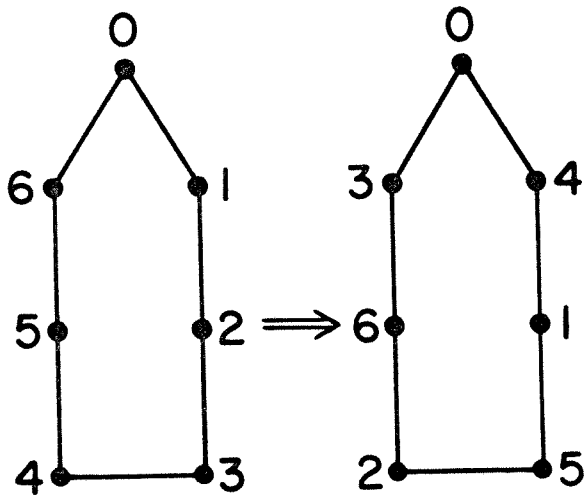
(b) The expected value of $t_A(\pi)$ satisfies the inequality

$$\bar{t}_A(\pi) \geq \frac{1}{16} n^2 - \frac{3}{16} n.$$

To prove this, let items i,j be located at u(i), u(j). Define $d(i,j) = \min\{|u(i)-u(j)|, n - |u(i)-u(j)|\}$ as the distance between items i and j. For any given permutation of the items π, define

$$I(\pi) = \sum_{i=0}^{n-1} d(i, i+m),$$

where i+m means (i+m) mod n. $n = 2m+1$. For the identity permutation π_o, $I(\pi_o) = nm$. For the permutation $\hat{\pi}(i) = 2i \mod n$, $I(\hat{\pi}) = n$.



$\hat{\pi}$ for n=7

Figure 12

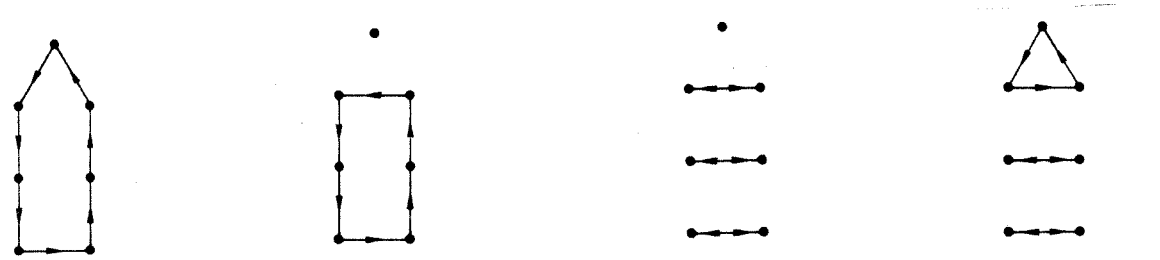
Note that application of a does not change the value of I, and application of e can change the value of I by at most 4. This is because e increases the distances between the item at 0 and those at 1,2,...,m-1 by 1, while decreasing the distances between it and those at m+1,...,n-2 by 1. Similarly, e increases the distances between the item at n-1 and those at m+1,...,n-2 by 1, while decreasing the distances between it and those at 1,2,...,m-1 by 1. In fact, after e only distances involving the items at 0 or n-1 may change. Therefore, I can be changed by at most 4.

To generate permutation $\hat{\pi}$, I has to be reduced from nm of π_o to n of $\hat{\pi}$, thus $\frac{1}{4}(nm-n) = \frac{1}{4}[\frac{n^2}{2} - \frac{3}{2}n]$ operations are necessary.

For the average case, the mean value of d(i,j) for any given permutation is $\frac{m+1}{2}$. Thus the expected value of I is $n(\frac{m+1}{2})$ and

$$\bar{t}_A \geq \frac{1}{4}\left[mn - n\left(\frac{m+1}{2}\right)\right] = \frac{1}{16}n^2 - \frac{3}{16}n.$$

More recently, another set of simple operations has been implemented. They are diagramatically as follows:



OPERATION a OPERATION b OPERATION c OPERATION d

Figure 13

Then the number of operations to access an item at location k and to maintain the ordering as described before can be cut down. For example, for $n=7$, $k=5$, the previous optimal algorithm (with only operations a and e) takes 5 operations (eae^2a) while here three operations are sufficient (cad).

In general, the optimal sequence of operations can be described as follows:

(1) For $1 \leq k \leq m$, apply $a^{k-1}\, d\, b^{k-1}\, c$. (Total = $2k$).

(2) For $m+1 \leq k \leq n-2$, apply $c\, a^{n-k-1}\, d\, b^{n-k-2}$. (Total = $2(n-k)-1$).

(3) For $k = n-1$, apply cac. (Total = 3).

To generate arbitrary permutations, one can devise an algorithm utilizing this optimal sequence as done before. It takes $O(n^2)$ operations for both the worst and average cases. The previous lower bounds hold here also.

Another important feature of magnetic bubble memory is its stopability, i.e., we can stop the rotation of the loop and even reverse its direction of rotation at any desirable moment without too much loss in time. We can then consider the following simplified model: Loops are stacked up to form a drum-like storage device, where the tracks are not constrained to rotate in synchronism, i.e., some tracks can stop, while others can go forward or backward. For simplicity, just consider the case when stop and forward of individual tracks are available. Assume that there are m tracks containing n records each. If we can have one step look-ahead of the request string, i.e., if we know two consecutive requests r_1, r_2 at a time, then we can rotate the track containing r_1 to the I/O port, while shifting the track containing r_2 optimally. The remaining tracks can either rotate with the track containing r_1 or remain intact.

If we assume random access request, these two options are statistically the same and the average access time per record is significantly reduced in comparison to the uncontrolled case [21].

Open Problem

(1) In the models used in the first part of this section, the access time is basically linear. Is it possible to build some other simple operations such that the access time can be cut down in an order-of-magnitude manner?

(2) Parallel to drum scheduling problems, we have now a whole new family of bubble memory scheduling problems to be solved.

(3) What is the role of queuing theory in all this?

REFERENCES

[1] C. Berge, The theory of graphs and its applications, Methuen & Co. Ltd., London, 1962.

[2] P. P. Bergmans, Minimizing expected travel time on geometrical patterns by optimal probability rearrangements, Information and Control, 20 (1972), pp. 331-350.

[3] A. K. Chandra and C. K. Wong, Worst-case analysis of a placement algorithm related to storage allocation, SIAM J. Computing, 4 (1975), pp. 249-263.

[4] R. A. Cody and E. G. Coffman, Record allocation for minimizing expected retrieval costs on drum-like storage devices, J. ACM, Vol. 23, No. 1, (1976), pp. 103-115.

[5] M. C. Easton and C. K. Wong, The effect of a capacity constraint on the minimal cost of a partition. J. Assoc. Comput. Mach., Vol. 22, No. 4, (1975), pp. 441-449.

[6] R. L. Graham, Bounds on multiprocessing anomalies and related packing algorithms, Proc. Spring Joint Computer Conf. 1972, pp. 205-218.

[7] D. D. Grossman and H. F. Silverman, Placement of records on a secondary storage device to minimize access time, J. Assoc. Comput. Mach., 20 (1973), pp. 429-438.

[8] G. H. Hardy, J. E. Littlewood, and G. Pólya, Inequalities, Cambridge University Press, Cambridge, England, 1952.

[9] D. S. Hirschberg and C. K. Wong, Upper and lower bounds for graph-diameter problems with application to record allocation, J. Combinatorial Theory, Series B, (to appear).

[10] R. M. Karp, A. C. McKellar, and C. K. Wong, Near-optimal solutions to a 2-dimensional placement problem, SIAM J. Computing, Vol. 4, No. 3, (1975), pp. 271-286.

[11] A. W. Marshall, I. Olkin, and F. Proschan, Monotonicity of ratios of means and other applications of majorization. In Inequalities, Shisha (Ed.), Academic Press, New York, 1967.

[12] A. C. McKellar and C. K. Wong, Dynamic placement of records in linear storage, (in preparation).

[13] S. E. Nevius, F. Proschan and J. Sethuraman, Schur functions in statistics. II: Stochastic Majorization. Florida State University Statistics Report M306, 1974.

[14] S. E. Nevius, F. Proschan and J. Sethuraman, Schur functions in statistics. III: A stochastic version of weak majorization, with applications. Florida State University Statistics Report M307, 1974.

[15] A. Ostrowski, Sur quelques applications des fonctions conveses et concaves au sens de I. Schur. J. Math. Pures Appl. 31 (1952), 253-292.

[16] F. Proschan and J. Sethuraman, Schur functions in statistics. I: The preservation theorem. Florida State University Statistics Report M254-RR, 1974.

[17] I. Schur, Uber ein Klasse von Mittlebildungen mit Anwendungen auf die Determinatentheorie. Sitzber. Berl. Math. Ges. 22 (1923), 9-20.

[18] C. K. Wong and D. Coppersmith, The generation of permutations in magnetic bubble memories, IEEE Trans. Comput., Vol. C-25, No. 3, (1976), pp. 254-262.

[19] C. K. Wong and A. C. Yao, A combinational optimization problem related to data set allocation, Special Issue on Computer Models and Performance, Revue Francaise d'Antomatique, d'Informatique, et de Recherche Operationnelle, 1976, (to appear).

[20] C. K. Wong and P. C. Yue, A majorization theorem for the number of distinct outcomes in N independent trials, Discrete Math. 6 (1973), pp. 391-398.

[21] C. K. Wong and P. C. Yue, The anticipatory control of a cyclically permutable memory, IEEE Trans. Comput., Vol. C-22, No. 5, (1973), pp. 481-488.

[22] P. C. Yue and C. K. Wong, On the optimality of the probability ranking scheme in storage application, J. Assoc. Comput. Mach., 20 (1973), pp. 624-633.

[23] P.C.Yue and C.K.Wong, Near-optimal heuristics for an assignment problem in mass storage. Internat. J. Computer and Information Sciences, Vol.4,No.4, (1975), pp.281-294.