ADAPTIVE REORGANIZATION OF DISK CYLINDER STRUCTURES

Scott D. Carson
Paul F. Reynolds, Jr.

Computer Science Technical Report No. 85-20
August, 1985

Submitted to
ACM Transactions on Computer Systems

Adaptive Reorganization of Disk Cylinder Structures

SCOTT D. CARSON
PAUL F. REYNOLDS, JR.

University of Virginia

Models of disk I/O systems often assume that the pattern of requests for cylinders is uniform, and that requests are mutually independent. Neither assumption holds well in practice. As a result, a disk system may experience much more head movement than its designer expects.

This paper presents a technique for reordering the cylinders on a disk to achieve a reduction in expected seek time, based on observed cylinder request probability distributions. The technique allows the disk system to restructure itself over time, eliminating the need for careful layout at design time. Using data taken from a typical computer system, the restructuring technique is shown to yield significant savings in expected seek time.

Categories and Subject Descriptors: B.3.2 [Memory Structures]: Design Styles - *mass storage*; B.4.2 [Input/Output and Data Communication]: Input/Output Devices - *channels and controllers*; B.4.4 [Input/Output and Data Communication]: Performance Analysis and Design Aids - *formal models*; C.4 [Performance of Systems]: *performance attributes*; D.4.2 [Operating Systems]: Storage Management - *secondary storage devices*; D.4.2 [Operating Systems]: File Systems Management - *access methods*;

General Terms: Algorithms, Measurement, Performance

Additional Key Words and Phrases: Adaptive Disk Reorganization, Disk Cylinder Placement

1. INTRODUCTION

A typical disk input/output system consists of a series of rotating platters that are divided into areas known as cylinders. An assembly of read/write heads moves from cylinder to cylinder servicing requests. Requests arrive to the system at random times, and each request is for a random cylinder. A queueing mechanism stores requests when requests arrive faster than they can be serviced.

The process by which the "next" request is selected from the queue is known as *head scheduling* [5]. For most scheduling disciplines, such as SCAN [4] and Shortest-Seek-Time-First [11], the average time required to move the head from one requested cylinder to the next approaches zero as the number of requests per unit of time becomes infinite. Thus, we expect the throughput of the system to become large under heavy offered loads. Unfortunately, most systems are not characterized by heavy disk loads. In fact, excessive I/O load typically represents a flaw in the design of the computer system as a whole [2].

In this paper we investigate techniques for reducing the average seek time that are effective even when the load is light. The methods described are based on the idea of reordering the cylinders (i.e., moving the data and providing a mapping function) such that, when requests arrive to a lightly loaded system, the resulting sequence of head positions is *probabilistically optimal*. In other words, we rearrange the information on the disk so that

Authors' address: Department of Computer Science, University of Virginia, Charlottesville, VA 22903.

the movement of the head approximates an optimal schedule that could be planned if the requests were known in advance. The net effect is that the disk system appears to be considerably faster than it would be without such reorganization.

The paper describes two models of disk system behavior. The first model represents the motion of the head as a Markov process, and is based on measured conditional head transition probabilities. The second model assumes that the requests are mutually independent; it requires only that the steady-state head position probabilities be measured. The first model is the better approximation, but requires quadratic space; the second model gives reasonable performance increases while requiring only linear space.

Because the optimal cylinder reorganization problem is NP-complete, we present approximate optimization algorithms for each of the two models. The algorithms are shown to produce very good results using request traces taken from a typical UNIX™ system. The numerical results suggest that the concept of cylinder reorganization can be extremely useful in practical applications.

2. MOTIVATION

Most studies of disk system behavior assume that the distribution of requests for cylinders is uniform, and that cylinder requests are probabilistically independent. For several reasons, neither assumption holds in practice. Unfortunately, head scheduling disciplines that take advantage of this nonuniformity have minimal impact on performance, because the probability that more than one request is in the queue is typically small. Most requests are simply serviced in First-Come-First-Served order.

A good example is provided by the 4.2 Berkeley UNIX (4.2 BSD) system. In this system large disks are divided into several smaller, logical devices, each of which contains a filesystem. These filesystems are arranged so that data in individual files can be accessed quickly [14]. This may be an appropriate design criterion in some applications; however, this arrangement may lead to performance degradation in applications such as timesharing, where total system performance is most important.

Nonuniformity of cylinder requests arises in the 4.2 BSD system for two reasons. First, directory structures are accessed much more frequently than files. Since each directory references many files, it is reasonable to expect that a directory will be accessed many times for each time a particular file is accessed. Second, placing multiple, independent filesystems on one physical disk can lead to nonuniformity, because different filesystems may be accessed at different rates.

Interdependence of cylinder requests in the UNIX system arises because operations on particular filesystems are often related. For instance, a directory lookup operation typically precedes access to the data contained in a file. Because multiple filesystems share one physical disk, there are typically several groups of cylinders within which requests are interdependent, and between which requests are independent.

As an example, consider Figure 1. This graph shows the cylinder access probability distribution measured on a Fujitsu EAGLE disk drive, during a day of typical activity. The 842-cylinder physical disk is divided into five logical devices: a "root" filesystem, a "swap" area, and three "user" filesystems. These five logical devices are shown horizontally separated in the figure.

Although no interdependence is shown in this figure, it is clear that requests are not uniformly distributed across the cylinders. The most apparent feature of Figure 1 is the sharp peak at cylinder zero (the leftmost). This is the location of the "super block" of the root filesystem; it is the root of the entire filesystem tree and is accessed several orders of magnitude more frequently than the average. Other peaks reflect the location of directory information on the disk.

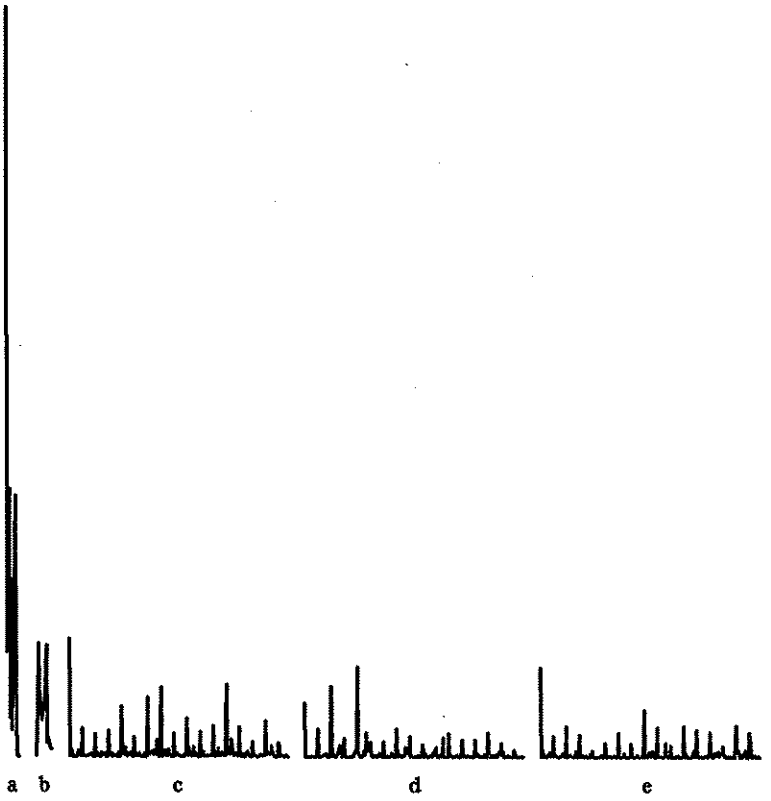


FIG. 1. Nonuniform cylinder access probability distribution measured on Fujitsu EAGLE disk with VAX 11/780 and 4.2 BSD UNIX (N = 842 cylinders). a. Root file system. b. Swap file system. c, d, e. User file systems.

3. BACKGROUND

Related studies have treated the seek time optimization problem in two ways. First, Maruyama and Smith [12] describe a method for reorganizing databases in which access probabilities for individual records are known. These records are allocated to particular cylinders in a way that minimizes seek time, balanced with the cost of performing insertions and reorganization operations. Their model is primarily concerned with performance optimization of individual files in database systems; it is much more microscopic than the model presented in this paper.

The second related model is that of Grossman and Silverman [9]. The Grossman and Silverman study is concerned with reorganizing disks on a cylinder-by-cylinder basis. The access probability distribution for individual cylinders is known, but the dependent probabilities are unknown. Grossman and Silverman divide disk accesses into two categories: purely independent accesses and purely sequential accesses. By combining the two access patterns in a Markov process, they obtain a reasonable approximation of disk system behavior.

Other related studies include those by Vaquero and Troya, which generalizes the results of Grossman and Silverman to include dependent probabilities [18], by Flory, Gunther, and Kouloumdjian, which describes clustering methods for improving seek time [6], and by Bergmans, which investigates reorganization techniques for linear (disk-like) and

other storage devices [1].

4. DEPENDENT PROBABILITY MODEL

The unique features of this study are that it takes account of dependent cylinder access probabilities, that it provides a means for adapting to changes in access patterns over time, and that it results in dramatic improvements in practical situations. In this section we describe the Markov model that is based on dependent cylinder access probabilities. We develop two heuristic algorithms for solving the optimal reorganization problem: a "greedy" algorithm and one based on the technique of *simulated annealing*.

4.1. Model

Let the disk contain N cylinders, numbered zero through $M = N-1$. Let the probability of accessing cylinder j next, given that the head is at cylinder i be p_{ij}. Further, let the steady-state probability that the head is at cylinder i be π_i. The π_i can either be measured directly or found as the eigenvalues of the matrix p_{ij}.

As long as the p_{ij} depend only on i and j, and not on any past history of head movement, then the motion of the head from cylinder to cylinder can be represented as a Markov process. In fact, it may be the case that the p_{ij} do depend on longer access patterns; the Markov model must be regarded as an approximation. Nonetheless, the results obtained with the Markov model are significant enough that the approximation appears to be a reasonable one.

As shown in [7], the expected seek distance $E[S]$, measured in cylinders, can be expressed as follows:

$$E[S] = \sum_{i=0}^{M} E[S \mid \text{head is at cylinder } i]\ prob(\text{head is at cylinder } i) \tag{4.1}$$

$$= \sum_{i=0}^{M} \sum_{j=0}^{M} |i-j|\ p_{ij}\ \pi_i .$$

If we let ρ_{ij} be the probability over all time of moving the head from cylinder i to cylinder j, then $\rho_{ij} = p_{ij}\ \pi_i$.

Now, we define an operator similar to the dot product operator, except that it operates on two N by N matrices. For any two N by N matrices $\bar{A}$ and $\bar{B}$ with elements a_{ij} and b_{ij}, $\bar{A} \odot \bar{B} \equiv \Sigma_{i=0}^{M} \Sigma_{j=0}^{M} a_{ij}\ b_{ij}$. The expected seek distance can now be expressed in terms of a matrix of costs (distances), and the matrix of head transition probabilities:

$$E[S] = \bar{c} \odot \bar{\rho}, \tag{4.2}$$

where

$$\bar{c} = \begin{bmatrix} 0 & 1 & 2 & \dots & M \\ 1 & 0 & 1 & \dots & M-1 \\ 2 & 1 & 0 & \dots & M-2 \\ \dots & \dots & \dots & \dots & \dots \\ M & M-1 & M-2 & \dots & 0 \end{bmatrix},$$

and

$$\bar{p} = \left[\rho_{ij} \right].$$

The problem of minimizing the expected seek distance based on the arrangement of cylinders becomes one of finding the permutation of either $\bar{c}$ or $\bar{p}$ that minimizes equation (4.2).[1] If we regard $\bar{p}$ as the set of probabilities of making transitions between *logical* cylinders (i.e., cylinder numbers as seen by the remainder of the system), then the permutation $\bar{c}'$ of $\bar{c}$ reflects the changed cost of making logical transitions due to cylinder reorganization. If we regard $\bar{p}$ as the set of probabilities of making transitions between *physical* cylinders (i.e., cylinder numbers as seen by the physical disk medium), then the permutation $\bar{p}'$ of $\bar{p}$ reflects the changed probability of making physical head transitions due to reorganization. For the remainder of this paper, we choose to take the latter view.

The Toeplitz form of the cost matrix $\bar{c}$ suggests the form of the (permuted) transition probability matrix that minimizes $E[S]$. In particular, $\bar{p}_{opt}$ will have its largest values near the diagonal, with values diminishing away from the diagonal. This explains two common file system design practices:

Place associated data near each other: If the transition from cylinder i to j is highly probable, then placing cylinder j next to (or near) cylinder i implies large values near the diagonal of $\bar{p}$. Since the values of $\bar{c}$ near its diagonal are small, this can be expected to lower the expected seek distance. This practice is used extensively in the 4.2 BSD UNIX file system [14].

Place the most frequently accessed records near the middle of the disk: If cylinder i is frequently accessed, then transitions to and from that cylinder may be more likely than others. Thus, the values of ρ_{ij} and ρ_{ji}, for any j, may be comparatively large. Since the values of $\bar{c}$ are smallest in the middle row and column, placing the large values of ρ near the middle lowers expected seek distance. This practice is used in the DEC FILES-11 filesystem, in which directory information is placed near the middle of each disk [19].

4.2. Algorithms

The combinatorial minimization problem described above gives rise to a decision problem that asks, for a given real number R, if there is a permutation of $\bar{p}$, $\bar{p}'$, such that $E[S] \leqslant R$. Let the minimization problem be called ρ_{MP}, and the decision problem be called ρ_{DP}.

THEOREM 1: ρ_{DP} is NP-complete.

PROOF: see [3].

It is shown in [3] that the problem ρ_{DP} is a generalization of the *Optimal Linear Arrangement* problem of circuit design theory [8]. In this problem, the objective is to find the linear arrangement of N interconnected circuits that minimizes the total length of wire required. The problem is also similar to the *Quadratic Assignment* problem, described in [8].

Because of the apparent intractability of ρ_{MP}, we seek appropriate heuristic methods for finding an approximation to $\bar{p}_{opt}$. In this section we consider two algorithms, both based on swapping adjacent cylinders.

The first algorithm is a "greedy" algorithm; it repeatedly examines each pair of adjacent cylinders and exchanges them whenever doing so reduces $E[S]$. Although the greedy algorithm is susceptible to becoming "stuck" at local optima, in practice it appears to

[1] A permutation of a matrix is a new matrix obtained by performing simultaneous row and column exchanges.

perform well.

Calculating the change in $E[S]$ for adjacent cylinder swaps is relatively simple. Let the cylinders to be swapped be cylinders γ and $\gamma+1$. The change in $E[S]$ can be expressed as follows:

$$\Delta E[S] = \text{new } E[S] - \text{old } E[S] \tag{4.3}$$

$$= E[S \mid \text{cylinder } \gamma \text{ is in location } \gamma+1] \tag{4.3a}$$

$$+ E[S \mid \text{cylinder } \gamma+1 \text{ is in location } \gamma] \tag{4.3b}$$

$$- E[S \mid \text{cylinder } \gamma \text{ is in location } \gamma] \tag{4.3c}$$

$$- E[S \mid \text{cylinder } \gamma+1 \text{ is in location } \gamma+1]. \tag{4.3d}$$

First, we note that swapping cylinders γ and $\gamma+1$ has no effect on terms of $E[S]$ that do not contain either γ or $\gamma+1$ as subscripts. Thus, all such terms of $\Delta E[S]$ cancel out, leaving $\Delta E[S]$ in terms of the "contributions" of cylinders γ and $\gamma+1$. The contribution to the old $E[S]$ by cylinder γ (4.3c) is

$$\sum_{i<\gamma} (\gamma - i)(\rho_{i\gamma} + \rho_{\gamma i}) + \sum_{i>\gamma} (i - \gamma)(\rho_{i\gamma} + \rho_{\gamma i}). \tag{4.4}$$

Likewise, the contribution to the old $E[S]$ by cylinder $\gamma+1$ (4.3d) is

$$\sum_{i<\gamma+1} (\gamma+1-i)(\rho_{i,\gamma+1} + \rho_{\gamma+1,i}) + \sum_{i>\gamma+1} (i - (\gamma+1))(\rho_{i,\gamma+1} + \rho_{\gamma+1,i}). \tag{4.5}$$

The contribution to the new $E[S]$ (4.3a) by cylinder γ, located at cylinder $\gamma+1$, is the expected distance traversed by the head during transitions to and from cylinder $\gamma+1$. This is just the sum over all cylinders of the probability of making transitions to and from γ times the distance from each cylinder to $\gamma+1$. Ignoring transitions between γ and $\gamma+1$ because the distance between γ and $\gamma+1$ remains unchanged, the contribution to the new $E[S]$ by γ is

$$\sum_{i<\gamma} (\gamma+1-i)(\rho_{i\gamma} + \rho_{\gamma i}) + \sum_{i>\gamma+1} (i - (\gamma+1))(\rho_{i\gamma} + \rho_{\gamma i}). \tag{4.6}$$

Finally, the contribution to the new $E[S]$ due to placing cylinder $\gamma+1$ in location γ is

$$\sum_{i<\gamma} (\gamma - i)(\rho_{i,\gamma+1} + \rho_{\gamma+1,i}) + \sum_{i>\gamma+1} (i - \gamma)(\rho_{i,\gamma+1} + \rho_{\gamma+1,i}). \tag{4.7}$$

Substituting (4.4) through (4.7) into (4.3) and rearranging terms gives the change in the expected seek distance due to swapping cylinders γ and $\gamma+1$:

$$\Delta E[S] = \sum_{i<\gamma} (\rho_{i\gamma} + \rho_{\gamma i}) - \sum_{i>\gamma+1} (\rho_{i\gamma} + \rho_{\gamma i}) - \sum_{i<\gamma} (\rho_{i,\gamma+1} + \rho_{\gamma+1,i}) + \sum_{i>\gamma+1} (\rho_{i,\gamma+1} + \rho_{\gamma+1,i}). \quad (4.8)$$

Equation (4.8) indicates that each cylinder seeks its own "center of probability". In other words, each cylinder tries to place itself in the center of the set of other cylinders that are likely to precede or follow it in the request stream. Cylinders that are accessed more frequently "dominate" those that are accessed less frequently, so after a number of swaps have taken place, each set of interdependent cylinders is arranged with the most frequently accessed cylinders in the middle of each set. At the same time, these sets of interdependent cylinders are moved toward each other.

The greedy reorganization algorithm is straightforward (see Figure 2). Each pair of adjacent physical cylinders is examined in sequence, and swaps are performed whenever $\Delta E[S]$ is less than zero. Whenever a swap is done, a table that maps "logical" to "physical" cylinders is updated. Although we do not show the details here, it is easy to see that the most significant computation in the algorithm, calculating $\Delta E[S]$, can be performed in O(1) time by storing the sums in Equation (4.8). These sums must be updated whenever cylinders are swapped.

```
Algorithm Greedy-ρ
repeat
   for γ ← 1 to M-1
      if ΔE[S] < 0 then
         swap cylinders (γ, γ + 1)
         update map
until satisfied

Algorithm Annealing-ρ
T ← T0
repeat
   γ ← uniform random integer between 0 and M-1
   if ΔE[S] < 0 then
      swap cylinders (γ, γ+1)
      update map
   else
      r ← uniform random real between 0.0 and 1.0
      if r < exp(−ΔE[S]/T) then
         swap cylinders (γ, γ+1)
         update map
   if sufficiently large number of swaps have been done then
      T ← kT
until frozen
```

FIG. 2. Greedy and annealing algorithms based on dependent accesses.

One difficulty arises in determining the stopping criteria. In our experiments we simply stopped after a sufficiently large number of swaps had been considered (10^6). While potentially inefficient, the results described in the next section show that the method works reasonably well.

The second algorithm is based on the process of *simulated annealing* [15] (Figure 2). Simulated annealing is a Monte-Carlo technique originally developed in the study of metals. The technique simulates the process of annealing, by which molten metals are cooled in a controlled fashion to yield a low-energy, stable structure. The application of simulated annealing to combinatorial optimization problems is shown in [10] to yield satisfactory results for problems such as traveling-salesperson and optimal wire-routing.

In the simulated annealing approach, a cylinder that might be swapped with its neighbor is randomly selected. If the move is a good one ($\Delta E[S] \leqslant 0$) then the swap is accepted. If not, then the swap is accepted with a probability based on the "current temperature" and $\Delta E[S]$. The temperature is initially high, so that much random movement occurs; this simulates the random motion of molecules in a molten metal. As the process continues, the temperature is lowered gradually so that random motion diminishes and a stable, near optimal configuration is reached; this simulates the forming of a stable, crystalline (low-energy) structure in the metal.

4.3. Experimental Results

To determine the effectiveness of the two algorithms, we applied them to cylinder trace data measured on a UNIX-based computer system. The results presented in this section show that cylinder reorganization can yield impressive performance improvements.

The computer system used for the tests is a Digital Equipment Corporation VAX 11/780. The system runs the 4.2 Berkeley version of the UNIX operating system, and serves a community of approximately 200 users. At any given time, 25 to 35 users are logged in. The disk drives are 400-megabyte Fujitsu EAGLEs.

The trace data were obtained by modifying the UNIX kernel to maintain a list of cylinders accessed for a particular disk drive. After a sufficiently large[2] number of accesses are recorded, the kernel stops saving requests and a user-level program reads the list from memory and saves it in a disk file. The process of recording accesses does not disturb the measurement; the disk file is written only after measurement is complete.[3] Each trace encompasses approximately one day.

Trace	Original Mean Seek Distance (cylinders)	Greedy Algorithm Mean Seek Distance	Greedy Algorithm Mean Seek (% of original)	Annealing Algorithm Mean Seek Distance	Annealing Algorithm Mean Seek (% of original)
1	110.4	39.6	36%	38.4	35%
2	75.5	30.2	40%	27.4	36%
3	141.1	40.9	29%	36.3	26%

FIG. 3. Table of results for dependent probability (ρ) algorithms.

[2] Our experiments each recorded 200,000 accesses.

[3] The measurement is disturbed slightly, in that when trace data are taken less space is available for I/O buffering in the kernel.

Results from three cylinder traces are summarized in Figure 3. Traces one and three represent days of typical user activity. Note that the final values of $E[S]$ are similar for both algorithms operating on these two traces. Trace two was measured while several large, computationally intensive programs were running; the swapping load on the system was more significant than usual. A lower final value of $E[S]$ was achieved for trace two because a small set of cylinders (those in the swapping area) were accessed frequently.

The results show that little benefit was achieved by using the slightly more complicated annealing algorithm. It may be the case that our trace data, by coincidence, did not produce the "local optimum" phenomenon to which the greedy algorithm is susceptible. Accordingly, we feel that the simulated annealing approach is the most appropriate, despite the small increase in computation time.

Figure 4 shows the cylinder access histograms for trace three. The three graphs represent (a) the original cylinder access histogram, (b) the histogram after the greedy algorithm has been applied, and (c) the histogram after the simulated annealing algorithm has been applied. Notice that both the greedy algorithm and the annealing algorithm move the large peaks toward the "center of probability". The greedy algorithm produces three pronounced groups of cylinders. The annealing algorithm produces the same three groups, but tends to move the groups closer together, resulting in a smaller final value of $E[S]$. Notice also that neither algorithm moves the largest peaks toward the physical center of the disk. This somewhat surprising result is explained by the observation that the position of the infrequently accessed cylinders that surround the large peaks makes little difference in $E[S]$.

5. INDEPENDENT PROBABILITY MODEL

Disk system models based on probabilistically independent cylinder accesses have been studied before. However, we present such a model here for two reasons. First, the independent model uses linear, rather than quadratic space; this may be more appropriate where disk reorganization is implemented in constrained environments, such as inside a disk controller. Second, we use additional techniques to improve the performance of the independent model; the disk is treated as a self-organizing list [13] with a cost function based on a specialization of Equation (4.8). Such techniques yield better results than might be expected from a purely independent model.

5.1. Model

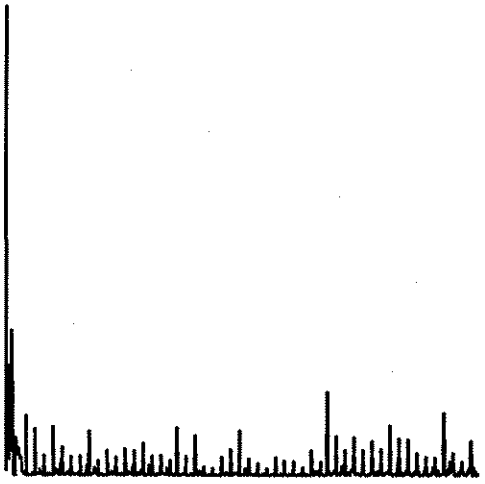
Suppose that requests are probabilistically independent. The probability that the next request is for cylinder j is independent of the current cylinder i; that is, $p_{ij} = \pi_j$, the steady state probability that the head is in position j. The long-term probability that the head moves from cylinder i to cylinder j is the probability that the head is at cylinder i, times the probability that the next request is for cylinder j:

$$\rho_{ij} = \pi_i \pi_j . \tag{5.1}$$

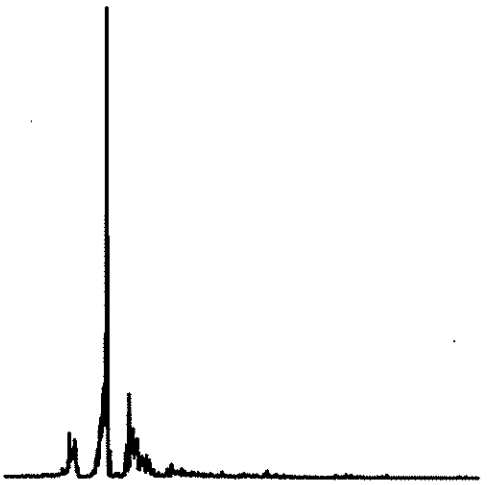
Now, suppose that we measure only the π_i. We can implement the Greedy-ρ or the Annealing-ρ algorithm by first computing $\Delta E[S]$, given the form of the ρ_{ij} in Equation (5.1), by substituting into Equation (4.8):

$$\Delta E[S] = \sum_{i<\gamma} (2\pi_i \pi_\gamma - 2\pi_i \pi_{\gamma+1}) - \sum_{i>\gamma+1} (2\pi_i \pi_\gamma - 2\pi_i \pi_{\gamma+1}),$$

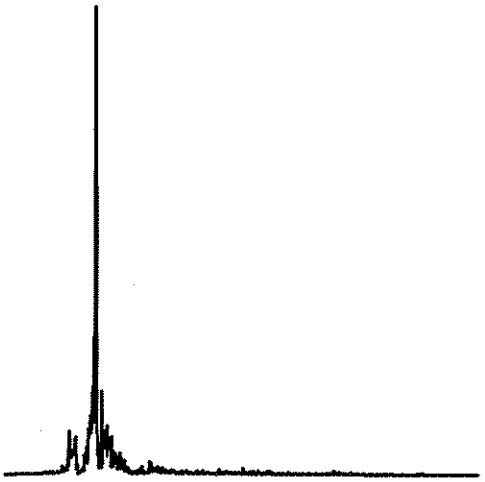
which reduces to



(a) Original distribution



(b) After application of Greedy-ρ algorithm



(c) After application of Annealing-ρ algorithm

FIG. 4. Cylinder access histograms for trace 3, ρ algorithms.

$$\Delta E[S] = 2(\pi_\gamma - \pi_{\gamma+1})(\sum_{i<\gamma} \pi_i - \sum_{i>\gamma+1} \pi_i). \quad (5.2)$$

Equation (5.2) indicates that exchanging the two cylinders, γ and $\gamma+1$, is beneficial when the more probable cylinder (of γ and $\gamma+1$) moves toward the more probable (right or left) set of other cylinders. Thus, Equation (5.2) attempts to place the most probable cylinders in the middle of the distribution, like Equation (4.8), but without clustering sets of related cylinders together.

5.2. Algorithms

Grossman and Silverman show in [9] that, given probabilistically independent cylinder accesses, the permutation of the matrix $\bar{\rho}$ (with elements $\pi_i \pi_j$) that minimizes $E[S]$ is found by placing the most frequently accessed cylinder in the middle of the distribution, with successively less probably cylinders placed on alternate sides. Thus, a polynomial-time algorithm for minimizing $E[S]$ exists; the problem is no longer NP-Complete.

Unfortunately, if the cylinder accesses are assumed to be independent, then the resulting minimization of $E[S]$ may have little or no correspondence with the actual system, in which accesses are not independent. In this section we present two algorithms that use the steady state position probabilities π_i, Equation (5.2), and the actual sequence of cylinder requests to mitigate this potential inaccuracy. The algorithms have one advantage over those presented in the previous section: since they require that only the π_i be measured, they can be implemented in linear space.

The first algorithm is a greedy one. A pair of cylinders, γ and $\gamma+1$, is selected, and $\Delta E[S]$ is computed as in Equation (5.2). If $\Delta E[S]$ is negative, then the two cylinders are exchanged.

The key feature of this algorithm is the method for selecting γ (and thus $\gamma+1$). Instead of allowing γ to iterate through the cylinders in increasing order, we iterate through the sequence of *requested* cylinders. If the position of the "next" cylinder is greater than (to the right of) the "current" cylinder, then we let γ be the current cylinder. If the position of the next cylinder is less than (to the left of) the current cylinder, then we let γ be the current cylinder minus one. This method implements a heuristic that moves two successively accessed cylinders (i.e., one after the other) closer together whenever doing so is beneficial ($\Delta E[S] < 0$). Thus, the heuristic is much like the *transpose* heuristic used for *self-organizing lists* [17].

The second algorithm is based on simulated annealing. In effect, it is the same as the Annealing-ρ algorithm presented in the last section, except that the choice of γ is made as in the Greedy-π algorithm, and that $\Delta E[S]$ is computed as in Equation (5.2). Both of the π algorithms are shown in Figure 5.

5.3. Experimental Results

The algorithms based on independent probabilities are inherently less powerful than those based on dependent probabilities, since they do not take into account the natural cylinder grouping discussed in Section 2. Nonetheless, our experiments indicate that with the addition of the transpose heuristic, the π algorithms compare favorably with the ρ algorithms discussed in Section 4. The results suggest that the π algorithms are of practical use when memory is at a premium.

The greedy and annealing algorithms were applied to the three measured cylinder traces described in the previous section. The results are summarized in Figure 6. As expected, the reduction in expected seek distance was less significant than with the ρ algorithms.

```
Algorithm Greedy-π
current cylinder ← first cylinder request
repeat
   next cylinder ← next cylinder requested
   if next cylinder > current cylinder + 1 then
      γ ← current cylinder
   else if next cylinder < current cylinder - 1 then
      γ ← current cylinder - 1
   if ΔE[S] < 0 then
      swap cylinders (γ, γ + 1)
      update map
   current cylinder ← next cylinder
until satisfied
```

```
Algorithm Annealing-π
T ← T0
current cylinder ← first cylinder request
repeat
   next cylinder ← next cylinder requested
   if next cylinder > current cylinder + 1 then
      γ ← current cylinder
   else if next cylinder < current cylinder - 1 then
      γ ← current cylinder - 1
   if ΔE[S] < 0 then
      swap cylinders (γ, γ+1)
      update map
   else
      r ← uniform random real between 0.0 and 1.0
      if r < exp(−ΔE[S]/T) then
         swap cylinders (γ, γ+1)
         update map
   if sufficiently large number of swaps have been done then
      T ← kT
until frozen
```

FIG. 5. Greedy and annealing algorithms based on independent accesses.

Trace	Original Mean Seek Distance (cylinders)	Greedy Algorithm Mean Seek Distance	Greedy Algorithm Mean Seek (% of original)	Annealing Algorithm Mean Seek Distance	Annealing Algorithm Mean Seek (% of original)
1	110.4	60.8	55%	45.9	41%
2	75.5	41.8	55%	38.8	51%
3	141.1	48.3	34%	43.3	30%

FIG. 6. Table of results for independent probability (π) algorithms.

It appears that the greedy algorithm became "stuck" at a local optimum when applied to trace one (Figure 6). With traces two and three the results of the greedy algorithm are slightly worse than those of the annealing algorithm. However, with trace one the greedy algorithm produced a larger final value of $E[S]$ than expected.

The cylinder access histograms for the two π algorithms are shown in Figure 7. Note the absence of pronounced groups of cylinders compared with Figure 4. This is a direct result of the way $\Delta E[S]$ is computed: the sole objective of Equation (5.2) is to place frequently accessed cylinders in the center of the distribution, with less frequently accessed cylinders surrounding them. Note also that the results of the greedy algorithm and the results of the annealing algorithm are nearly indistinguishable.

6. SEEK TIME CALCULATIONS

So far we have used seek *distance* as a minimization criterion, because the integer distances inherent in the disk mechanism are convenient algebraically. Seek *time* is an equivalent metric; the matrix $\bar{c}$ (Equation 4.2) is of the same form: symmetric, Toeplitz, and monotonic. In this section we briefly summarize the results obtained thus far, showing the improvements in seek time resulting from disk reorganization.

Seek time on modern disks, like the EAGLE, increases as a function of the distance between two cylinders (Toeplitz form). For very light head mechanisms, the acceleration and deceleration of the head become negligible[4], and seek time can be approximated accurately with a linear function. This function contains a constant term that reflects a constant delay in starting head motion. Thus, the seek time function, for any two cylinders i and j, is

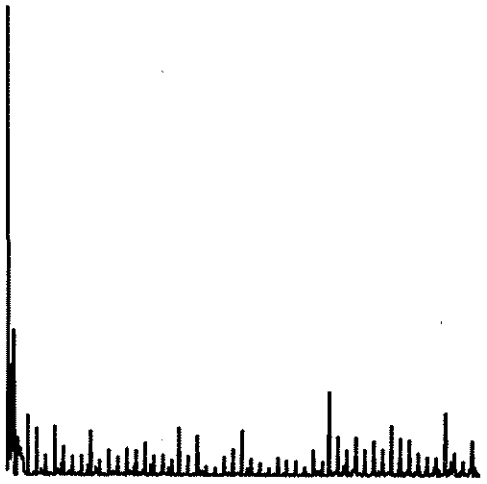
$$T(i,j) = |i - j|c_1 + c_2, \tag{6.1}$$

where c_1 and c_2 are parameters of the disk drive.

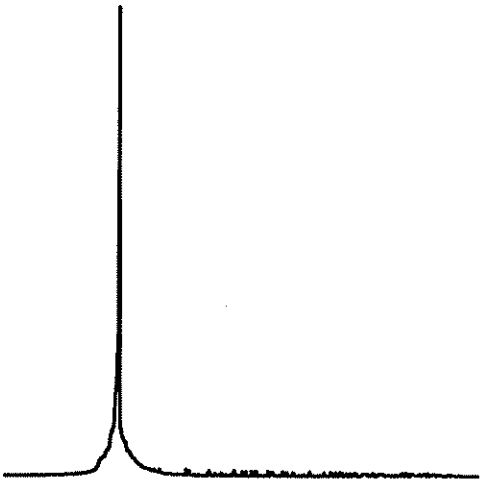
For the Fujitsu EAGLE disk drive, the single-track seek time is 5 milliseconds, and the worst-case seek time (cylinder 0 to cylinder 841) is 35 milliseconds [16]. This gives parameters $c_2 = 4.96$ milliseconds, and $c_1 = 0.0357$ milliseconds per cylinder.

Substituting the seek distance results from Figures 3 and 6 into Equation (6.1), we obtain the seek time results shown in Figure 8. Results are shown for both annealing and greedy versions of the two (ρ and π) algorithms. While the percent improvement in seek time is somewhat smaller than the improvement in seek distance, the results continue to indicate a significant benefit.

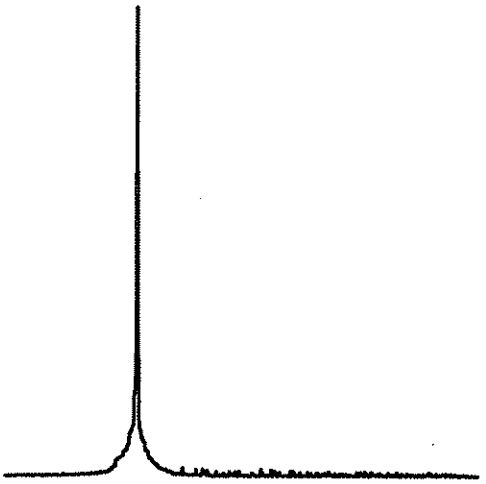
[4]More properly, the acceleration and deceleration are absorbed into the constant c_2.



(a) Original distribution



(b) After application of Greedy-π algorithm



(c) After application of Annealing-π algorithm

FIG. 7. Cylinder access histograms for trace 3, π algorithms.

Trace	Original Mean Seek Time (msec)	Greedy ρ Mean Seek Time/ % of Orig.	Annealing ρ Mean Seek Time/ % of Orig.	Greedy π Mean Seek Time/ % of Orig.	Annealing π Mean Seek Time/ % of Orig.
1	8.90	6.37/72%	6.33/71%	7.13/80%	6.60/74%
2	7.66	6.04/79%	5.94/78%	6.45/84%	6.35/83%
3	10.00	6.42/64%	6.25/63%	6.68/67%	6.51/65%

FIG. 8. Table of seek time results for all algorithms.

7. APPLICATIONS

The two disk reorganization methods discussed in this paper can be used in real computer systems to restructure disks over time. The different properties of the dependent and independent probability methods suggest that they are each useful in different applications. In this section we discuss ways in which the ρ and π algorithms might be applied. Additionally, we consider the problem of determining the length of time over which the algorithms should adapt.

The dependent-probability-based algorithms yield the best results, but require considerable space. Thus, they are only suitable for implementation on systems with large amounts of main (or virtual) memory. An implementation envisioned by the authors works as follows.

The "disk driver" is normally a part of the operating system kernel[5]. Thus, memory that it uses is typically not swappable; the corresponding physical memory is unavailable for any other use. It is often inappropriate to dedicate sufficient memory to store either the $\bar{\rho}$ matrix or a trace of disk accesses. Thus, some other method must be used.

One possible implementation consists of a queue of disk accesses (cylinder numbers) that is maintained by the disk driver. The queue can be read from kernel memory by a user-level program. The user-level program executes the Annealing-ρ algorithm, storing the $\bar{\rho}$ matrix in virtual memory. Once a near-optimal cylinder configuration has been obtained, the user program formulates a sequence of cylinder exchanges that results in the final configuration, and requests that the disk driver perform the swaps.

The independent-probability-based algorithms are somewhat easier to implement. Since both π algorithms require only a histogram of cylinder accesses (and thus only linear space), the π algorithms can be executed directly by the disk driver. The stream of incoming cylinder requests drives the algorithm, as shown in Figure 5. The reorganized cylinder configuration is maintained as a permutation, and at during some period of inactivity the permutation is applied to the physical cylinders.

The simplicity of the π algorithms suggests that they are suited to a wide variety of applications. For example, the π algorithms might be implemented within a disk *controller*, resulting in a physical device that adapts itself to access patterns independently of the computer system. Additionally, existing software disk drivers can be simply retrofitted for adaptive reorganization.

One question that arises involves selecting an appropriate period of time over which the algorithms are applied. The data sets that drive the experiments reported in this paper were each measured over the course of one day. Since the access distributions differ from day to day, it may be that the reorganization process must be applied over a longer period to yield good results. The danger in selecting an adaptation period that is too short is that access patterns of the recent past might be a poor indication of those of the near future.

[5]This is true of all UNIX systems.

Preliminary investigation has shown that, while cylinder access patterns do change from day to day, patterns observed over several days maintain a non-uniform shape, with some cylinders accessed several orders of magnitude more frequently than others. Although more research is needed before conclusive results can be stated, it appears that the reorganization techniques discussed in this paper will yield significant savings in seek time when applied over longer periods of time.

8. CONCLUSION

This paper discusses two approaches to solving the NP-complete Optimal Disk Cylinder Arrangement problem: a direct method based on dependent probabilities and an heuristic method that uses independent probabilities. The results obtained thus far suggest that further research in this area is promising. The authors are currently pursuing two research directions.

The first area of research involves investigating other heuristics for solving the reorganization problem. The fundamental problem is related to several wire-routing and matrix manipulation problems. It remains to be seen whether other methods for solving NP-complete problems, such as "divide and conquer," might be applied to this problem.

Another research topic that deserves further attention is the interaction between head scheduling disciplines and disk reorganization. This study does not take queueing into account. Instead, we compare mean seek distances for FCFS schedules, before and after cylinder reorganization. Under the SCAN discipline that UNIX uses, it is reasonable to expect that the seek distance falls when queueing takes place with or without cylinder reorganization. Of course, reorganization makes queueing less likely, so the process of adapting to request patterns while head scheduling takes place is not straightforward.

The results presented in this paper show that disk performance can be significantly improved by changing the location of information on the disk. The study differs from others in that it describes an adaptive method that is based on cylinder access patterns. Further, the paper describes practical algorithms that are effective, yet easy to implement on modern computer systems.

ACKNOWLEDGEMENT

The authors would like to acknowledge their useful discussions with James Cohoon, who first suggested simulated annealing as a solution technique, and with Dana Richards, who noticed the parallel between disk reorganization and self-organizing lists.

REFERENCES

1. BERGMANS, P. P. Minimizing expected travel time on geometrical patterns by optimal probability rearrangements. *Inf. and Control 20* (1972), pp. 331-350.

2. BRANDWAJN, A. Personal communication.

3. CARSON, S. D., AND REYNOLDS, P. F. The NP-completeness of the optimal disk cylinder arrangement problem. Computer Science Technical Report No. 85-13, University of Virginia, April, 1985.

4. COFFMAN, E. G., KLIMKO, L. A., AND RYAN, B. Analysis of scanning policies for reducing disk seek times. *SIAM J. Comput. 1*, 3 (September 1972), pp. 269-279.

5. DENNING, P. F. Effects of scheduling on file memory operations. Proc. AFIPS 1967 Spring Joint Comput. Conf., Vol 31, pp. 9-21.

6. FLORY, A., GUNTHER, J., AND KOULOUMDJIAN, J. Data base reorganization by clustering methods. *Inform. Syst. 3* (1978), pp. 59-62.

7. FRANK, H. Analysis and optimization of disk storage devices for time-sharing systems. *J. ACM 16*, 4 (October 1969), pp. 602-620.

8. GAREY, M. R., AND JOHNSON, D. S. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Co. San Francisco, 1979.

9. GROSSMAN, D. D., AND SILVERMAN, H. F. Placement of records on a secondary storage device to minimize access time. *J. ACM 20*, 3 (July 1973), pp. 429-438.

10. KIRKPATRICK, S., GELATT, C. D., AND VECCHI, M. P. Optimization by simulated annealing. *Science 220*, 4598 (May 1983), pp. 671-680.

11. TEOREY, T. J., AND PINKERTON, T. B. A comparative analysis of disk scheduling policies. *Commun. ACM 15*, 3 (March 1972), pp. 177-194.

12. MARUYAMA, K. and SMITH, S. E. Optimal reorganization of distributed space disk files. *Commun. ACM 19*, 11 (November 1976), pp. 634-642.

13. MCCABE, J. On serial files with relocatable records. *Oper. Res. 13 (1965), pp. 609-618.*

14. MCKUSICK, M. K., JOY, W. N., LEFFLER, S. J., AND FABRY, R. S. A fast file system for UNIX. *ACM Trans. Comput. Syst.* 2, 3 (August 1984), pp. 181-197.

15. METROPOLIS, N., ROSENBLUTH, A. ROSENBLUTH, M., TELLER, A., AND TELLER, E. Equation of state calculations by fast computing machines. *J. Chem. Phys. 21*, 6 (June 1953), pp. 1087-1092.

16. M2351A/AF Mini-Disk Drive CE Manual. Fujitsu America Inc., Santa Clara, CA.

17. RIVEST, R. On self-organizing sequential search heuristics. *Commun. ACM 19*, 2 (February 1976).

18. VAQUERO, A. AND TROYA, J. M. Placement of records on linear storage devices. *Proc. IFIP Cong.* (1980), pp. 331-336.

19. VAX/VMS DCL Dictionary. Digital Equipment Corp., Maynard, MA, p. DCL-323.