DATA STRUCTURES AND ACCESS METHODS FOR READ-ONLY OPTICAL DISKS

Eduardo Fernandes Barbosa[1] and Nivio Ziviani[2]

[1] Departamento de Engenharia Eletrônica
[2] Departamento de Ciência da Computação
Universidade Federal de Minas Gerais
Belo Horizonte, Brazil

ABSTRACT

This paper presents alternatives for efficient file organization and placement on read-only optical disks, known as CD-ROM. We extend an analytical model for retrieval costs found in the literature, adding additional aspects to the model, such as rotational latency and file structure. From CD-ROM standards we derive algorithms for file mapping and access time calculations to build a deterministic model which is used in simulations. Experimental results on different file structures and file sizes show that rotational latency, file size, file structure, and file allocation strongly affects the retrieval performance in CD-ROM disks. A comparative study considering a sequence of accesses to sequential, indexed sequential, B-tree and hashing file structures shows that hashing and indexed sequential are the organizations that presented the smallest total time under the limitations of optical devices. We present an indexed sequential organization for CD-ROM file systems which minimizes disk accesses.

INTRODUCTION

Optical disk technology is currently being used as secondary storage devices with significant advantages over magnetic medium, such as higher storage capacity and lower cost. Read-only compact disks, referred as CD-ROM, is presently being used as storage device for large data base distribution. The growing interest on CD-ROM is due to its large storage capacity (up to 600 megabytes), low cost to the end user and high level of physical and logical integrity. The main differences between CD-ROM and magnetic disks are: (i) CD-ROM is a read-only medium and, as such, the structure of the information is static; (ii) the efficiency of data access is affected by its location on the disk and by the sequence in which it is retrieved. (iii) due to constant linear velocity (CLV) the tracks have variable capacity and rotational latency depends on disk position.

The CD-ROM track has a spiral pattern and an access to distant tracks requires more time due to the need of optical head displacement and changes in disk rotation. The capability of accessing nearby tracks with no displacement of the reading mechanism is called *span* and the number of tracks which can be accessed in this way is called *span size*. In actual CD-ROM drives span size is up to 60 tracks. Accessing data located within span boundaries, seek time takes only 1 millisecond per additional track, which is performed by a single angular laser beam displacement from the current track, which is called *anchor point*. In this situation, seek time is negligible compared to rotational latency. When accessing data located outside span boundaries, seek time requires from 200 to 600 milliseconds due to optical head displacement.

Rotational latency is smaller on inner tracks as a consequence of the constant linear velocity (CLV) scheme adopted in CD-ROM. The smaller rotational latency takes 60 milliseconds and the larger one takes 138 milliseconds. So, file allocation on outer tracks effectively reduces the number of head displacements for any sequence of data accesses because the file occupies a smaller number of tracks (up to 2.3 times smaller), but, conversely, the final cost is charged to larger rotational latencies in this disk area.

For the purposes of this paper, we define an *object* as a record of the data file or the index file. A *qualifying object* is one that should be retrieved to answer a given query. To answer a query it might be necessary to retrieve more than one object. Any object is supposed to be completely contained within a track (called *non-crossing objects*), and the smallest object size that can be retrieved is one sector large (2048 bytes).

Given a sequence of objects to be retrieved an scheduling algorithm provides an ordered sequence of anchor points that define the head displacements necessary to answer a given query. There are different sets of anchor points and schedules that may answer a query that needs two or more qualifying objects. Tracks within a span may overlap with another span from a neighboring anchor point, that is, there may be a subset of the tracks that belongs to both spans.

Laub (1986) and Zoellick (1986) describe the evolution of optical disks and compare them to magnetic disks. Design considerations can be found in Lambert (1986), Ropiequet (1987) and Sherman (1989). Christodoulakis (1987, 1988 and 1989) presents an analytical model for CD-ROM cost estimation to compute the expected number of optical head displacements needed to answer a query. These papers establish fundamental concepts for CLV optical disk modeling and help to understand the main issues related to CD-ROM logical design. Some early results were presented in Barbosa e Ziviani (1991).

The objective of this paper is to analyze the problem of efficient file organization and placement on CD-ROM disks. We extend the model presented by Christodoulakis by considering file structure and rotational latency for real cost estimation. We present a comparative study for sequential, indexed sequential, B-tree and hashing file structures showing that hashing and indexed sequential are the best choices under the limitations of optical devices. We show how to minimize disk accesses to indexed sequential files by presenting an algorithm and an index structure that uses in full extent the span capability of the CD-ROM drive.

AN EXTENDED ANALYTICAL MODEL FOR CD-ROM

The analytical model proposed by Christodoulakis (1987, 1988, 1989) allows cost measurements by giving the expected number of head displacements that the access mechanism has to perform for the retrieval of a random sequence of objects from CD-ROM disks. The model is developed under a probabilistic approach, taking into account the file size, track capacity and number of objects for retrieval. An important contribution of this model is the definition of an optimal scheduling for retrieving data. These conditions proved to minimize the number of times that the access mechanism is moved, as well as the total distance traveled by the optical head. The theorem for optimal scheduling is stated as: The number of head displacements required to access all qualifying objects in a query is minimized if (a) spans of two consecutive anchor points do not overlap and (b) an anchor point is always positioned on a track with qualifying objects. Moreover, the total distance traveled by the access mechanism is also minimized if the optical head is moved in only one direction during the retrieval of a set of objects (in addition to conditions (a) and (b)). The proof can be found in Christodoulakis (1987).

Although the model gives optimal results with respect to the expected number of head displacements necessary to answer a given query, it ignores the contribution of rotational latency in the retrieval costs. The contribution of rotational latency is ignored by making the assumption that seek time is much larger than rotational latency. We extend this model showing that this assumption is not correct in general, that is, rotational latency plays an important role in many situations.

Following, we present analytical results showing the influence of rotational latency and file structure to estimate the actual cost retrieval, by extending the model presented by Christodoulakis (1987, 1988). The approximated solution for the expected number of optical head displacements H_d to retrieve non-crossing objects with no span overlapping, obtained by Christodoulakis (1987, 1988, p. 290) is

$$H_d = \frac{B}{1 + \frac{Q-1}{N_t-1}(B-1)} \tag{1}$$

where Q represents the span size, N_t represents the number of tracks of the file and B represents the expected number of tracks with objects that qualify a query, which is given by

$$B = N_t \left[1 - \frac{\binom{C \times N_t - C}{n}}{\binom{C \times N_t}{n}} \right] \tag{2}$$

where C is the track capacity, given by the number of objects in each track, and n is the number of objects to be retrieved.

The influence of rotational latency in the retrieval costs can be obtained as follows:

1. Compute the total time spent with head displacements to retrieve n objects;
2. For each retrieved object, add the time due to rotational latency.

We consider also the influence of file structure in the retrieval cost by taking into account l levels of addressing, where l depends on the file structure. Thus, the total access time to retrieve and transfer n objects is given by

$$t_a = H_d \times \overline{t_s} + n(1 + l)(\overline{t_r} + b_{tt}) \tag{3}$$

where $\overline{t_s}$ is the average seek time for each head displacement, $\overline{t_r}$ is the average rotational latency, b_{tt} is the block transfer time to main memory, and $l = 0$ for direct address files and $l > 0$ for indexed files with l index levels residing on disk.

The average time to retrieve one object is given by

n	H_d Head Displacement		$\overline{t}_a$ (ms) No Latency		$\overline{t}_a$ (ms) With Latency	
	Inner tracks	Outer tracks	Inner tracks	Outer tracks	Inner tracks	Outer tracks
500	257	146	270	160	336	325
1000	346	170	186	99	251	264
1500	391	180	143	73	209	239
2000	418	186	118	60	183	225
2500	435	189	100	51	165	217
3000	448	190	88	45	153	211
3500	457	192	79	41	144	207
4000	464	193	71	38	136	203
4500	470	194	66	35	131	201
5000	475	194	61	33	125	199

Table 1: Average access time with and without rotational latency as a function of n

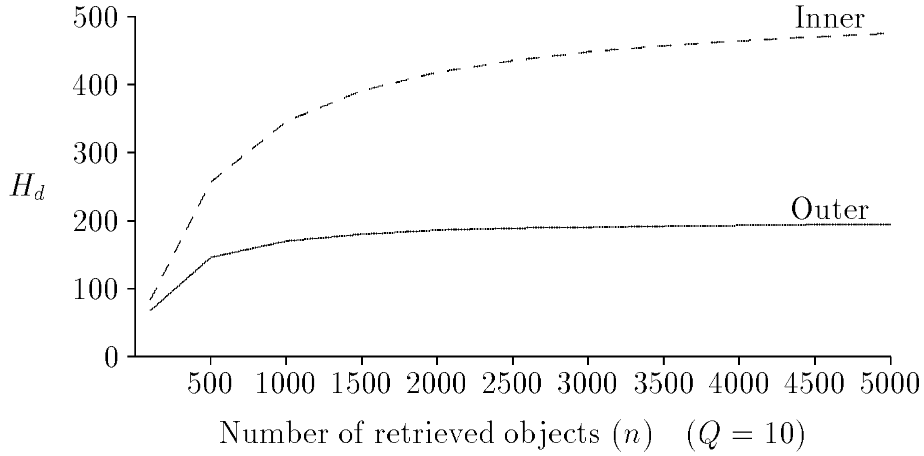

Figure 1: Head displacements according to Table 1

$$\overline{t_a} = \frac{t_a}{n} \tag{4}$$

Table 1 presents data related to a file with 45,000 sectors (90 megabytes), using Eq. (1) and Eq. (4), span size $Q = 10$ and the number n of retrieved objects in the range of 500 to 5,000. Notice that a 45,000 sectors file occupy 5,000 internal tracks and 1,956 external tracks. The plot in Figure 1, related to columns two and three of Table 1, illustrates the difference in the number of head displacements H_d for equal sized files allocated on the inner and outer tracks. This is the example used by Christodoulakis (1988, pp. 293–294) to support a general claim that frequently and randomly access files should be allocated on outer tracks. Following, we present a different result for this example by considering the influence of rotational latency.

We extend his example by adding the time due to rotational latency in the last two columns of Table 1. The data related to average access time $\overline{t_a}$ without latency (columns three and four) was obtained from the data presented in columns two and

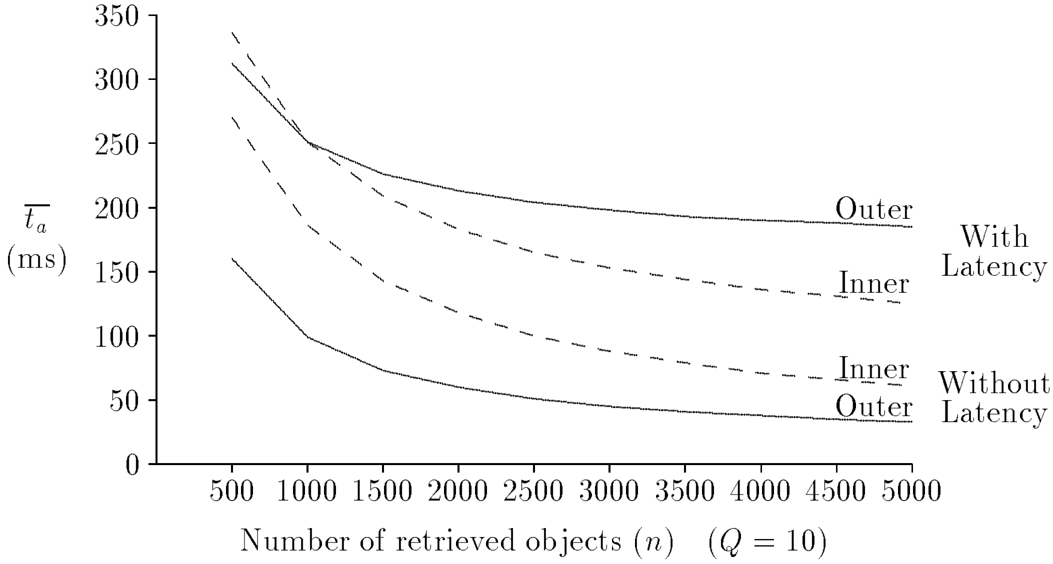

Figure 2: Average access time with and without rotational latency

three by multiplying each value by $\overline{t_s}$ ($\overline{t_s} = 500$ milliseconds), plus the time for one block transfer ($b_{tt} = 13.33$ milliseconds). The data related to average access time $\overline{t_a}$ with latency was obtained by using Eq. (4), considering $\overline{t_r} = 65$ milliseconds for inner tracks, $\overline{t_r} = 153$ milliseconds for outer tracks and $l = 0$ (direct access files).

The plot in Figure 2, related to the last four columns of Table 1, shows that if the time due to rotational latency is considered then the average time to retrieve one object $\overline{t_a}$ is smaller when the file is located on the inner tracks.

The above discussion leads us to the conclusion that frequently and randomly accessed files using the optimal scheduling algorithm should be allocated on inner tracks. Another example that clearly illustrates this conclusion is the following: if a file can be completely contained within span boundaries in any position on disk then retrieval costs will be smaller if this file is located on inner tracks, where rotational latency is smaller.

In practice the objects are not searched according to the optimal scheduling algorithm: the references to objects in the file are done at random. Even in this situation, experimental data presented in Section "Experimental Results" confirm the above results for large and very small files. As we are going to present, there are few exceptions for some small files in the range of 1 to 3 megabytes, which might present smaller average access time when allocated on outer tracks. This is because such file sizes tend to require very few head displacements, when span size is close to the typical value of 60 tracks.

FILE MAPPING AND ACCESS TIME EVALUATION

A deterministic model for file mapping into tracks and sectors is developed from CD-ROM standards. The model is developed from physical and logical characteristics of the disk. The physical characteristics considered were the spiral pitch of the tracks (standard value is 1.6 micrometer), their dimensions, angular velocity, linear velocity (standard value is 1.3 meter/second), sector size and time to read a sector (standard value is 13.33 milliseconds). The logical characteristics considered were sector capacity (2048 bytes), transfer rate (150 kbytes/second), disk capacity, and minimum and maximum track capacity (standard values are 9 and 20 sectors per track, respectively).

The model permits to map files into tracks and sectors, providing exact access time results for any retrieval condition imposed to the reading device. It permits: obtaining the capacity of a given track (in sectors); finding the track corresponding to any given sector address; determining the latency time for any given track; determining initial and final sectors and tracks corresponding to a span size in any disk position; obtaining the number of head displacements necessary to move to the next sector address; and finally, determining the access time to locate sectors and tracks from the drive specifications. This deterministic model was used in a CD-ROM simulator to obtain exact values for access times. The simulator was used to obtain the experimental results presented in Section "Experimental Results".

Disk Mapping

Track Capacity. The capacity for the i^{th} track is given in sectors by

$$C_i = C_1 + (i-1)\,\delta S \tag{5}$$

where C_1 is the track capacity (in sectors) of the first track and δS is the difference between any track and the next. We know that the first track has $C_1 = 9$ sectors and the last has $C_n = 20.6$ sectors. Also, for these boundaries, we know that the number of tracks is $n = 22,500$, and therefore $\delta S = 5.155 \times 10^{-4}$ (sectors). Although the CD-ROM track has a spiral pattern, each complete turn might be approximated by a perfect circle, due to the very small pitch between tracks, and the speed of each track might be also considered as constant.

Track Corresponding to a Given Sector. Firstly, we need the total number of sectors, N_s, from track 1 to track n, obtained from Eq. (5), as follows:

$$N_s = (n \times C_1) + \delta S \sum_{i=1}^{n-1} i \tag{6}$$

The track number corresponding to a given sector address is obtained by solving Eq. (6) for $n = N_t$, as follows:

$$N_t = \frac{\delta S - 2C_1 + \sqrt{(2C_1 - \delta S)^2 + 8N_s\,\delta S}}{2\,\delta S} \tag{7}$$

From the above expression it is possible to compute the track number related to a given sector address, which is necessary to evaluate the seek time. The amount of data contained between two given tracks can also be derived. As an example, the volume of data contained on the 10 outermost tracks is approximately 2.3 times greater than the volume of data that can be contained on the 10 innermost tracks of the CD-ROM. The combination of Eq. (5) and Eq. (7) permits to identify any sector of a given track.

Evaluation of Span Coverage. The volume of data, in kilobytes, contained in a span coverage is given by:

$$BW = \frac{C_i + C_j}{2} \times Q \times 2 \tag{8}$$

where Q is the span size, C_i is the track capacity for track i and C_j is the track capacity for track j, both obtained from Eq. (5), where i and j are the track numbers which

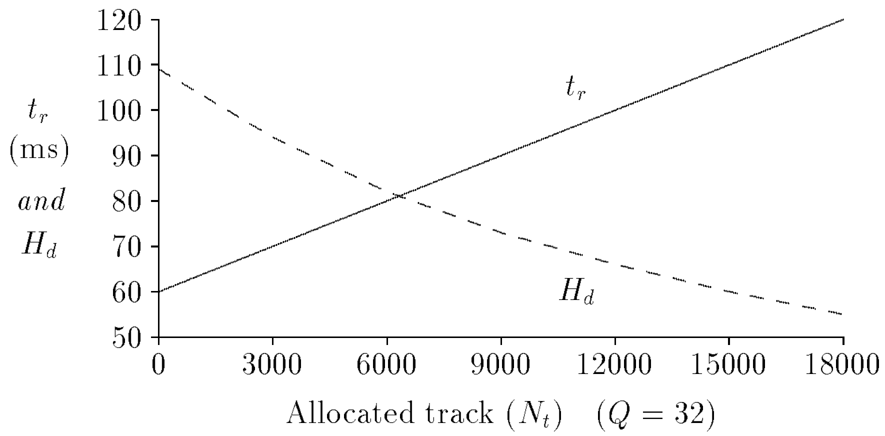

Figure 3: Rotational latency and head displacement as a function of file allocation

bounds a span, that is, $i = a$, the first track of the span with anchor point in track a, and $j = a + Q - 1$, the last track of the same span.

Sector coverage depends not only on the span size, but also on the optical head position over the disk. For inner tracks span coverage is smaller than for outer tracks, as a consequence of differences in track capacity. For a span size $Q = 64$ tracks, the first 64 tracks (inner tracks) are covered from the anchor point allocated in track 1, and the span coverage BW is 1.154 megabytes. The last 64 tracks (outer tracks) are covered from the anchor point allocated in track 22,437. At this location, the span coverage is 2.6346 megabytes. The ratio between these two capacities is 2.3 times. This characteristic demonstrates the importance of file distribution on the disk, since the volume of data covered on the outer tracks is much larger than inner tracks. Conversely, on outer tracks the rotational latency is larger. Thus, the relevant question is: What is the best file placement to minimize access time? In Section "Experimental Results", where real applications are simulated (and so the optimal scheduling algorithm does not apply), it will be shown that the best file placement will depend on the file size and file structure.

Figure 3 shows the number of head displacements H_d needed for retrieve all records of a 12.8 megabytes file (25,600 objects of 512 bytes each), with nonoverlapping spans, as a function of file allocation on disk. In this example, the span size was set to 32 tracks. For comparison purposes, it is also shown the variation of rotational latency as a function of track number.

Access Time Evaluation

The *access time* (t_a) to retrieve an object is given by:

$$t_a = t_s + t_r + t_x \tag{9}$$

where t_s is the *seek time*, t_r is the *rotational latency time*, t_x is the *transfer time* to move data from disk to main memory. We consider that t_s depends on the drive's technology and head displacement, t_r depends on the track radius and t_x depends on the amount of data to be retrieved in each disk access. All the references to data is done on the basis of sector addressing, which can be easily converted from sector number to the format *minute*:*second*:*sector*.

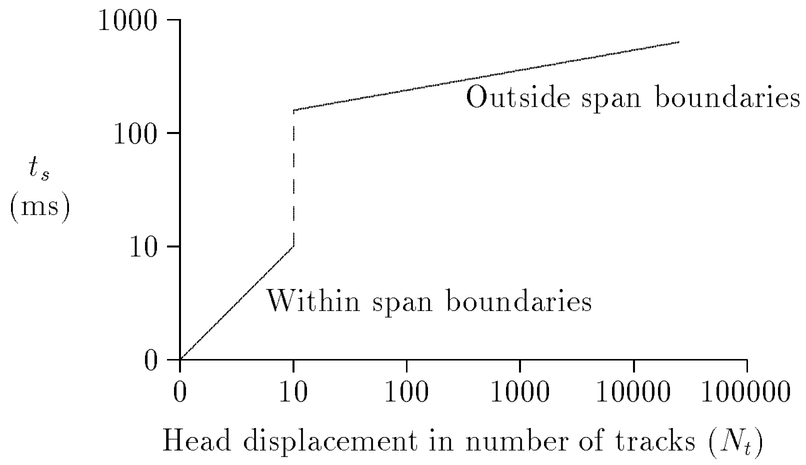

Figure 4: Seek time for a reading device ($Q = 10$)

If the next track to be accessed is within span boundaries then t_s is computed by considering the displacement of the laser beam at a rate of 1 millisecond per track. This figure is negligible, even when compared with the smallest rotational latency or seek time for tracks allocated outside the span boundaries. If the next track is outside span boundaries, typical values of t_s are between 200 and 600 milliseconds. This time is due to the optical head displacement and its settling in the new position (focusing and tracking adjustments plus rotational stabilization). Seek time for tracks outside span boundaries depends on the distance that must be traveled by the access mechanism. Figure 4 presents typical seek time as function of head displacement magnitude for a reading device with span size of 10 tracks.

Computation of rotational latency is done by considering track capacity and the time required to read one sector (which is the same for any track). Thus, $t_r = (C \times b_{tt})/2$, where C is defined by Eq. (5) and b_{tt} is the block transfer time. Typical values of t_r are between 60 and 138 milliseconds.

Transfer time is found by multiplying the number of sectors to be read (block size B) by the block transfer time and so, $t_x = B \times b_{tt}$. Typical transfer rate is 150 Kbytes per second for any position on disk, implying that the time to read one sector is 13.33 milliseconds.

FILE ORGANIZATION AND ACCESS METHODS

In this section, we present a comparative study of data structures and access methods for CD-ROM. The B-tree, sequential, hashing and indexed sequential file organizations are compared considering the restrictions and characteristics of the CD-ROM medium. It is shown that file organizations and algorithms suitable for dynamic applications, usually adequate for traditional magnetic medium, are not necessarily the best choices for read-only disks. We present an indexed sequential organization that minimizes the number of disk accesses, by providing an index structure that enables the reading device to access the maximum number of records within span boundaries.

B-Trees

B-tree structure is very good for dynamic applications, with provision for frequent additions and deletions of records. The expected storage utilization is 69% (Knuth, 1973). Due to the static characteristic of the information stored in CD-ROM media there is no reason to keep empty spaces for future use. Consequently, B-Tree files are not among the best choices for CD-ROM disks.

Sequential Files

If the application accesses data sequentially then the sequential file organization is the most efficient for CD-ROM, both in terms of access time and space utilization. The only existing limitation is the transfer ratio from disk to memory. Some kind of data is always stored and retrieved in sequential form, such as audio and video files.

For small files the sequential organization is also adequate for random access to data. For example, the average access time for a 100 Kbytes file is 330 milliseconds after the access mechanism has been positioned at the beginning of the file.

Hash Files

Hashing file organization is highly efficient since it requires only one disk access for each data retrieval (it requires at most one head displacement and only one rotational latency). Ideally, different keys should map to different addresses, but this is not always possible, and two or more keys may hash to the same address. Such an occurrence, called collision, can be reduced by applying some techniques, such as reducing the storage density, increasing bucket size, and choosing a hash function that evenly spreads all the keys in the available addresses. As a disadvantage, this structure does not allow sequential access to records. When this factor is not important, hashing is the organization method of choice.

Indexed Sequential Files

Indexed sequential files make possible both random and sequential access. The indexed sequential structure is implemented in magnetic disks by keeping a cylinder index in main memory. In this scheme, each data access is done with only one seek, since each cylinder contains a block index with the highest key value on each block in that cylinder index. For dynamic applications this steady state condition can not be kept if a large number of records is added to the file. In the case of CD-ROM, this organization is particularly interesting because of the static nature of the information.

An Efficient Indexed Sequential File for CD-ROM. Indexed sequential structures can be efficiently implemented in CD-ROM, by considering the static nature of the information and the span capabilities of the reading device. The set of tracks covered by a span in a CD-ROM might be compared to the set of tracks belonging to a cylinder in a magnetic disk. We consider each set of tracks bounded by a span as an *optical cylinder*. Two differences between the two media are: the optical cylinders have different capacities (depending on disk position) and a subset of tracks might belong to two or more optical cylinders (span overlapping).

Motivated by this fact and recalling that accessing times depend on disk position we propose an indexed sequential organization for CD-ROM. In this file organization, the reading mechanism will be positioned at pre-selected optical cylinders, avoiding span

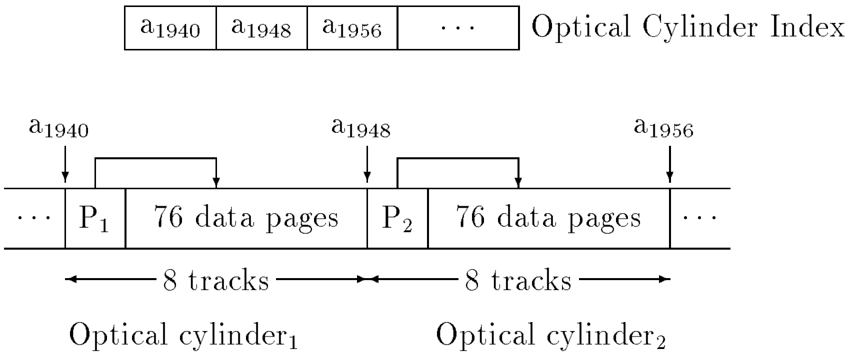

Figure 5: Diagram of the indexed sequential file organization proposed

overlapping. This will be accomplished by designing an index structure so that each index page refer to the largest number of data pages allocated within span boundaries. The diagram in Figure 5 shows this file organization for a 3 megabytes file starting at track number 1940, where each data page is 1 sector long (a span size of 8 tracks starting in track 1940 occupies approximately 78 sectors). To build it we do as follows:

1. Allocate the file on disk by computing initial and final tracks;

2. Compute the total number of optical cylinders needed to cover all file tracks without overlapping, and find their respective anchor points;

3. Build a *Optical Cylinder Index*: It contains the highest key value associated to records stored in the tracks covered by each span (this index should be kept in main memory).

4. Build a *Page Index*: It contains the highest key value associated to records in each data page, comprising the actual span coverage. This page should be stored in the track selected as anchor point, defining an optical cylinder. Due to static characteristic of the information and its regular structure, the explicit addressing might be avoided by computing the address of the next node at lower level from the current node.

Any record is retrieved as follows: given a search key, the corresponding optical cylinder (span) and its anchor point is obtained from the *Optical Cylinder Index*. The reading mechanism is then moved to the selected anchor point (this is the only long seek operation involved). The *Page Index* is then read and the data page containing the given search key is found within the actual span boundaries. This structure were built considering the two conditions for optimal scheduling proposed by Christodoulakis (1987). Here spans of consecutive head displacements do not overlap and the access mechanism is always positioned in an optical cylinder with qualifying objects.

Total access time is given by the sum of the times to find the anchor point, to read the index page and to read the data page, as follows

$$t_a = (t_s + t_r + P_i \times b_{tt}) + (t_r + P_d \times b_{tt}) \qquad (10)$$

where P_i is the index page size and P_d is the data page size.

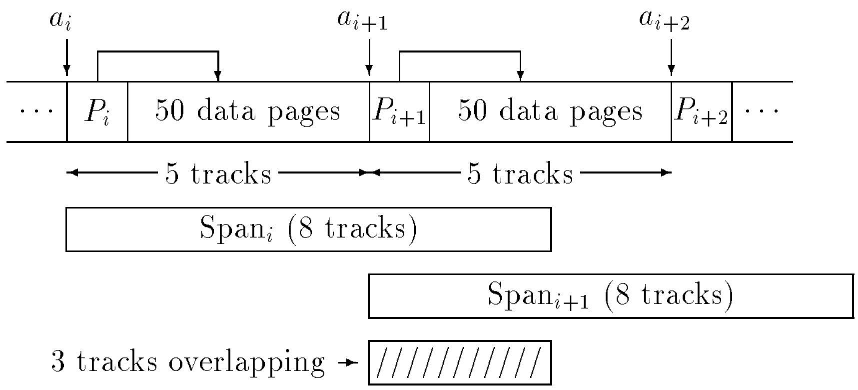

Figure 6: Example of span overlapping in indexed sequential files ($Q = 8$)

The example shown in Figure 6 illustrates one kind of span overlapping. Let us consider a segment of a file allocated in a disk region where track capacity is 10 sectors and data page and page index are 2 Kbytes (one sector) long. For an index entry of 40 bytes, each page index P_i might refer to $\lfloor 2048/40 \rfloor = 50$ data pages. This corresponds to 50 sectors or 5 tracks in the given conditions. Notice that index pages will always be anchor points and two adjacent anchor points a_i and a_{i+1} will be separated by 5 tracks. If span size is 8 tracks, then we have an overlapping of 3 tracks.

Figure 7 presents a similar problem. If the number of data pages referred to by an index page is larger than the span coverage then there is a chance of having to perform one or more seeks to retrieve a set of records referred to by pointers in the same index page. For a file segment with the same characteristics as the one described in Figure 6 but considering span size $Q = 3$, from an anchor point it is not possible to reach all the objects referred to by the current index page and a new head displacement might be necessary. Moreover, span overlapping also occurs, a worse situation in terms of performance as compared with the example of Figure 6.

The two problems just described can be eliminated by using an page index size so that one page index and all related data pages occupy exactly the number of tracks belonging to the span size Q. This is exactly the indexed sequential structure we have proposed above. In fact, experimental results show that this is the situation when both total access time t_a and number of head displacements H_d present best results. The worst result is obtained when the number of data pages referred to by an index page is larger than the span coverage, as illustrated by Figure 7.

Christodoulakis (1989) has proposed a modified B-Tree structure, named BIM-Tree (Balanced Implicit Multiway Tree). This structure is a perfect balanced m-ary tree, similar to a modified B-Tree in which nodes are completely full. This structure is also similar to the indexed sequential structure proposed above. The main difference is that our algorithm distributes data into pages in a way to fully exploit the span access capability of the optical reading device avoiding span overlapping to obtain reduction of head displacements.

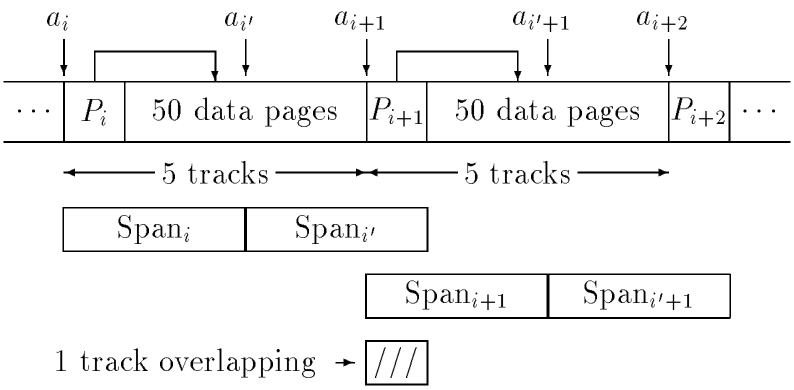

Figure 7: Example of another kind of span overlapping in indexed sequential files ($Q = 3$)

EXPERIMENTAL RESULTS

The CD-ROM simulator based on the deterministic model described in Section "File Mapping and Access Time Evaluation" was used to simulate several files with different structures and sizes. The simulator is informed about file size, block (or page) size, file location, index page size (if applicable), record size, reading device parameters and the number of search keys. It outputs the file mapping, access times for simulated queries and some statistics, such as average, best and worst cases. It also enables the user to perform variations on data file structure and reading device parameters (within boundaries), so that it is possible to examine the influence of these changes on retrieval performance for a given file, using a particular reading device.

The simulator considers actual time characteristics for t_s, as found in actual CD-ROM drives, according to the plot shown in Figure 4. The objects for retrieval may be selected according to the optimal scheduling algorithm (Christodoulakis, 1987) or to a random scheduling algorithm. In a random scheduling each object to be retrieved is selected from a random key generator, each key being converted into sector and track number, depending on the file structure.

Table 2 presents analytical and experimental results for the optimal scheduling algorithm, related to a file with 45,000 sectors (90 megabytes), occupying the 4,500 inner tracks with average track capacity of 10 sectors per track and span size $Q = 10$. This is the same example used in Section "An Extended Analytical Model for CD-ROM" (see Table 1), extracted from Christodoulakis (1988, pp. 293–294), but considering an average track capacity of 10 sectors per track (instead of 9 sectors per track considered by Christodoulakis, which is the capacity of the first track of the file). Obviously, the number of head displacements (H_d) for the optimal scheduling algorithm is smaller than any other algorithm. We can see from Table 2 that for large numbers of objects to be retrieved (n), the number of head displacements for random scheduling increases considerably, mainly due to span overlapping. The simulation results are obtained within 95% confidence interval.

The experimental results presented in the next subsections were obtained using actual CD-ROM drives parameters and random scheduling. Some general conclusions

n	H_d		
	Optimal Scheduling		Random Scheduling
	Analytical	Experimental	Experimental
100	83	83 ± 0.050	100 ± 0
500	244	244 ± 0.045	499 ± 0
1000	322	320 ± 0.050	997 ± 0
2000	383	380 ± 0.050	1996 ± 0
3000	408	405 ± 0.048	2993 ± 0
4000	422	417 ± 0.050	3992 ± 0
5000	430	426 ± 0.038	4985 ± 0

Table 2: Analytical and experimental results for head displacements (H_d)

for randomly and directly accessed files that maybe derived from this data are:

- Files up to 800 Kbytes can be completely contained within the typical span size of 60 tracks, and so require no head displacements. In that case, retrieval costs will be smaller if the file is allocated on inner tracks, were rotational latency is smaller.

- Very large files (250 megabytes or more) are weakly affected by disk location. Such large files fills up to nearly 50% of the disk, and the number of head displacements needed to retrieve a given number of objects is practically the same in any disk position. Also, average rotational latency is not considerably different for large files when allocated on outer or inner tracks.

- Intermediate sized files (1 to 3 megabytes) present better performance when allocated on outer tracks. Such files, when allocated on inner tracks, overflows the boundaries of span coverage leading to head displacements; when allocated on outer tracks, the files are completely contained within span coverage, leading to no head displacements at all.

In the next subsections we present simulation results for sequential, hashing and indexed sequential files ranging in size from 0.1 to 204.8 megabytes, allocated from the first track to the last one, using different sizes for data and index blocks. During simulations we also considered the influence of different drive parameters, including the span size. Simulation results were presented in terms of best, worst and expected access time and the number of head displacements, as a function of track position and span size. Each result represents the average over 1,000 retrievals. More details can be found in Barbosa (1990).

Sequential Files

The plot in Figure 8 presents simulation results for a 160 Kbytes sequential file with span size $Q = 64$. Some observations are:

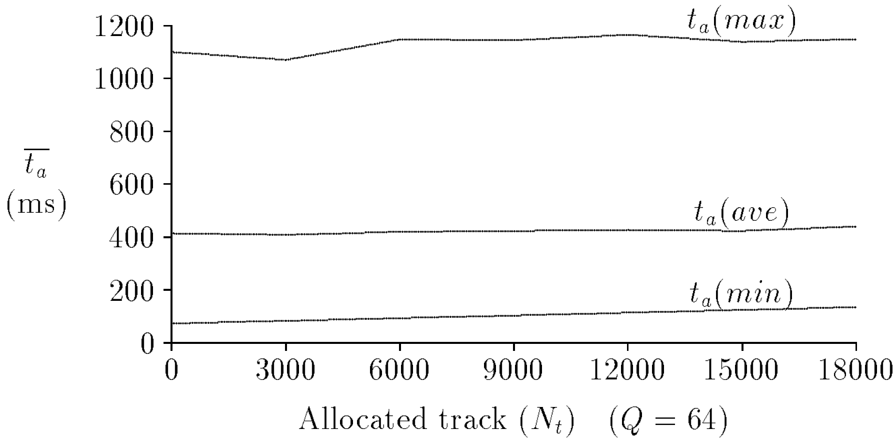

Figure 8: Access time for a sequential file 160 Kbytes large

(a) For minimum access ($t_a(min)$) times there are no head displacements or sequential readings because only the first record of the file is read (only rotational latency and transfer time are involved). In this case, the times are directly proportional to the file position on disk (in Figure 8, for track 1, $t_a = 73$ ms; for track 18,000, $t_a = 135$ ms).

(b) Maximum access times are not affected by file position, because this figure depends only on the time for sequential reading of the file and this time is constant for any position on disk, since the transfer rate is the same for any track. Similarly, expected access times are not affected by file position on disk.

Consequently, sequential file structures are not affected by file position on disk.

Hash Files

The plots in Figures 9, 10 and 11 present simulation results for hash files. Some observations are:

(a) Hash files present the smallest average access times when compared to sequential and indexed sequential, since any object is retrieved with only one disk access and one rotational latency.

(b) The plot in Figure 9 shows that for very small files the best allocation is on inner tracks. Since there is no head displacements for that file size, for the same reasons described in Section "Sequential Files", access times are directly proportional to file position on disk.

(c) The plot in Figure 10 shows results for a representative of intermediate sized files, in the range of 1 to 3 megabytes. It can be seen that they present better performance when allocated on outer tracks, for the reasons presented in Section "Experimental Results". The plot in Figure 11 shows that very large files are weakly affected by file position on disk (variation less than 10%).

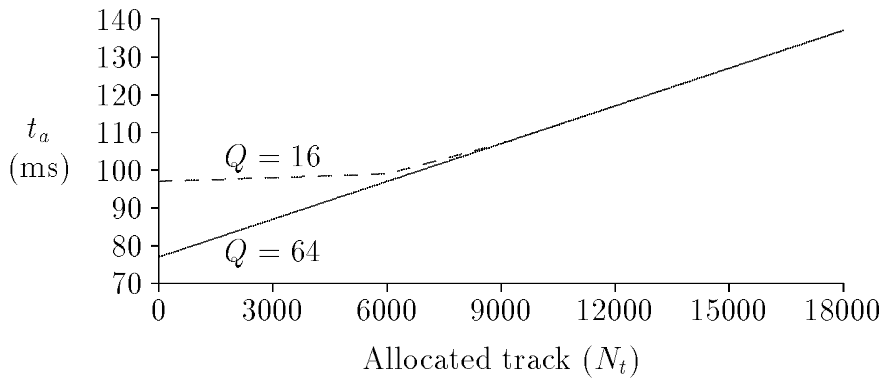

Figure 9: Access time for a 0.2 Mbytes hash file

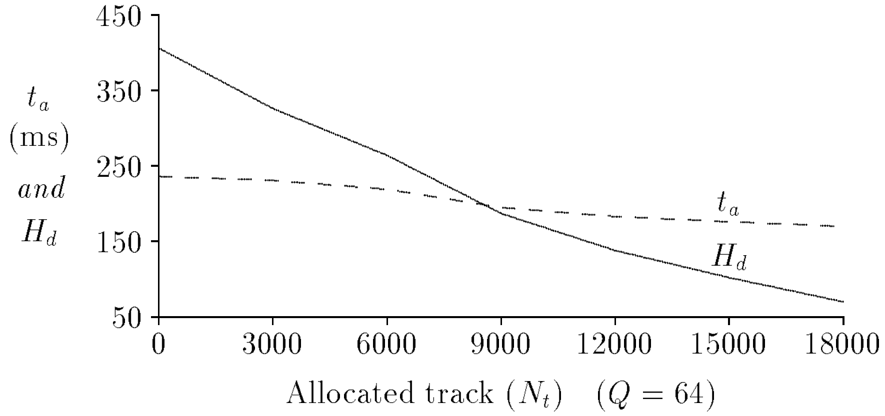

Figure 10: Head displacements and access time for a 1.6 Mbytes hash file

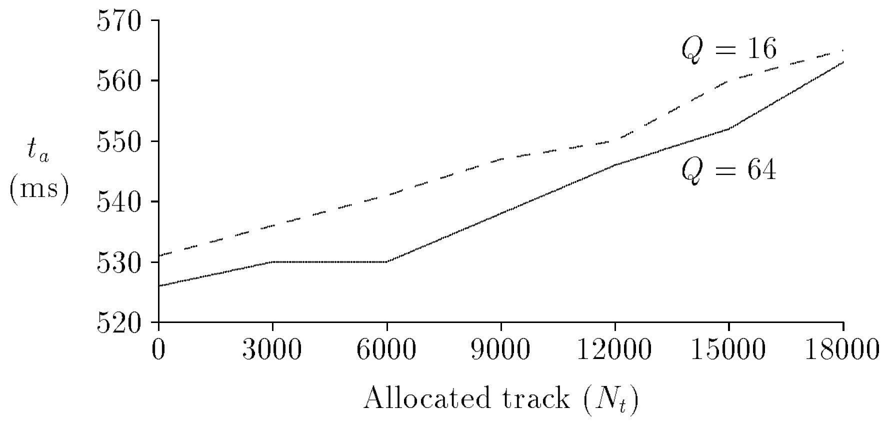

Figure 11: Access time for a 102.4 Mbytes hash file

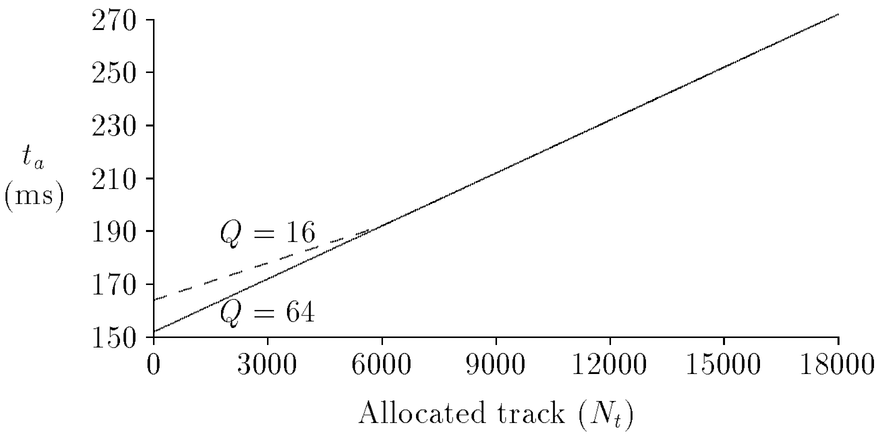

Figure 12: Access time for a 0.2 Mbytes indexed sequential file

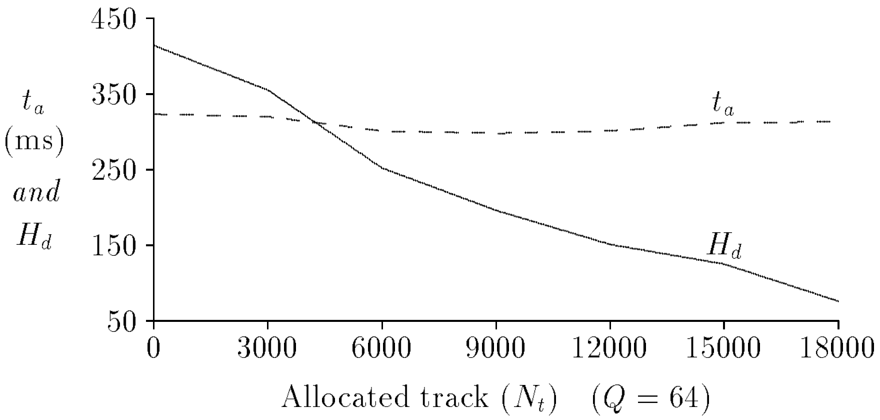

Figure 13: Head displacements and access time for a 1.6 Mbytes indexed sequential file

Indexed Sequential Files

Figures 12, 13, 14, 15 and 16 show simulation results for indexed sequential files. It can be seen the effects of file size, file allocation, and span size on the retrieval performance. Analysis of these results show that:

(a) The plot in Figure 12 shows that for very small files the best allocation position is on inner tracks.

(b) The plot in Figure 13 shows that for intermediate file sizes the number of head displacements decreases as the file is allocated on outer tracks and the smallest access times are obtained when the file is allocated on intermediary tracks. This is because indexed sequential files require at least two rotational latencies, one to retrieve the index page and one to retrieve the data page.

(c) The plot in Figure 14 clearly shows that the best file allocation is on inner tracks. The plot in Figure 15 shows that very large files are weakly affected by file position on disk and also by variations in span size.

(d) The plot in Figure 16 shows that the larger is the span size the least is the influence in retrieval costs.

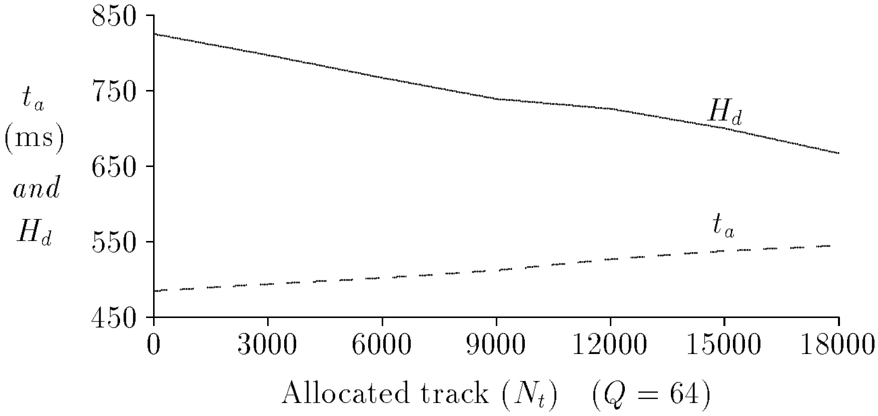

Figure 14: Head displacements and access time for a 6.5 Mbytes indexed sequential file

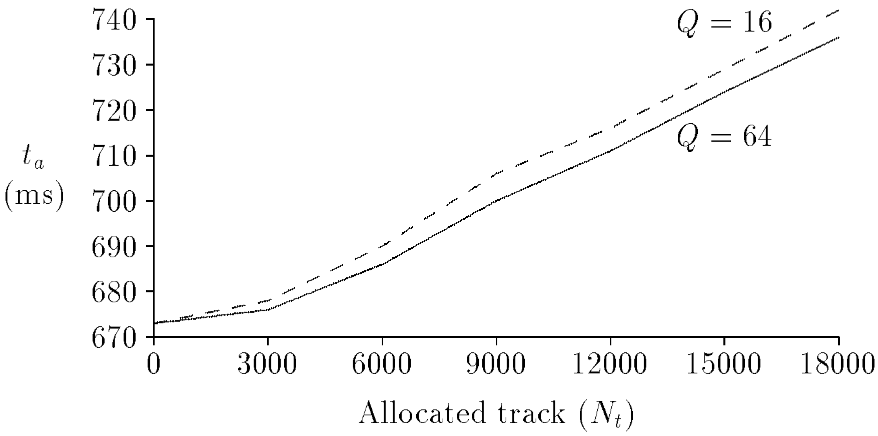

Figure 15: Access time for a 204.8 Mbytes indexed sequential file

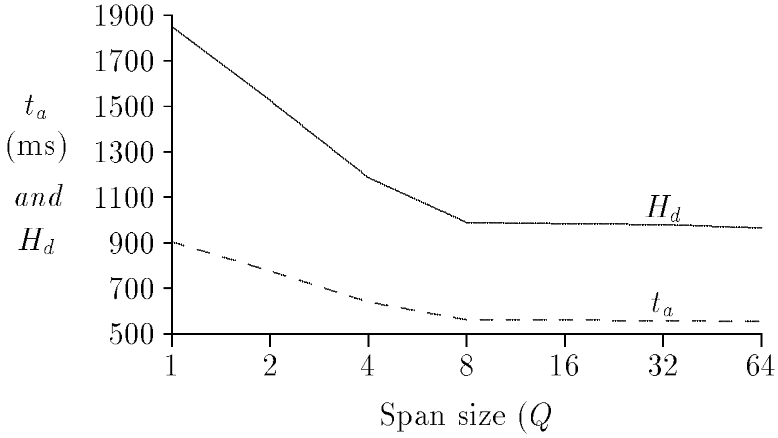

Figure 16: Head displacements and access time for a 26 Mbytes indexed sequential file

SIMULTANEOUS PLACEMENT OF FILES

An interesting optimization problem is the simultaneous placement of files with different sizes and structures. Some files performances are strongly affected by disk position while others are position independent. The problem is to find an optimal placement for all files which minimizes overall retrieval costs.

We propose the following heuristic: Given n files to be allocated on m available tracks ($n \ll m$), for each file we determine the difference between maximum and minimum costs. This difference can be obtained from simulation by varying the file position from the first to the last track. Next, we sort the files by decreasing order of the difference between maximum and minimum costs and allocate them in this order.

This criteria is based on the fact that files with larger differences between maximum and minimum costs will affect more intensily the total retrieval cost and, therefore, they should be the first to be allocated on disk. Consequently, the last files to be allocated will be those with smaller differences in cost for any track. In other words, after having allocated the most critical files, the remaining disk space should be used to allocate files that are weakly affected by disk position, such as sequential files.

CONCLUSIONS

This paper emphasizes the main influence of CD-ROM features in design strategies for organizing and placing files using this technology. We discuss the analytical model proposed by Christodoulakis (1987) and show the importance of rotational latency in considering the computation of retrieval costs. We also show that file size, file structure and file placement on disk have decisive influence on retrieval performance in CD-ROM. A deterministic model was developed from parameters of commercially available CD-ROM drives. This model allows file mapping and the exact computation of access time for various retrieval conditions imposed to the reading device. This enables us to perform a comparative study of B-Tree, sequential, hash and indexed sequential files. Considering specific features and limitations of CD-ROM environment, we concluded that indexed sequential and hash are the best organization choices for register structured files. We present an indexed sequential organization for CD-ROM file systems which minimizes disk accesses. Experimental results obtained from simulation confirm the extended model presented in this paper, leading to basic designing rules on CLV read-only optical disks.

ACKNOWLEDGMENTS

The authors wish to acknowledge the many fruitful discussions with Márcio L. B. de Carvalho. Financial support from the Brazilian CNPq - Conselho Nacional de Desenvolvimento Científico e Tecnológico and IBM do Brasil are also gratefully acknowledged.

REFERENCES

Barbosa, E. F. (1990) "Estruturas de Dados e Alocação de Arquivos em Discos CD-ROM", *Master Thesis*, Departamento de Ciência da Computação, Universidade Federal de Minas Gerais, Brazil.

Barbosa, E. F. and Ziviani, N. (1991) "Estruturas de Dados e Métodos de Acesso para Discos Óticos de Apenas-Leitura", *VI Simpósio Brasileiro de Bancos de Dados*, Manaus, Brazil, 39–58.

Christodoulakis, S. (1987) "Analysis of Retrieval Performance for Records and Objects Using Optical Disk Technology", *ACM Transaction on Database Systems 12* (2), 137–169.

Christodoulakis, S. and Ford, D. A. (1988) "Performance Analysis and Fundamental Performance Trade Offs for CLV Optical Disks", *Proceedings of ACM SIGMOD*, 286–294.

Christodoulakis, S. and Ford, D. A. (1989) "File Organizations and Access Methods for CLV Optical Discs", *ACM SIGIR Forum 23* (1 & 2), 152–159.

Knuth, D.E. (1973) *The Art of Computer Programming; Volume 3: Sorting and Searching.* Addison-Wesley.

Lambert, S. (1986) CD-ROM *The New Papyrus*, Microsoft Press.

Laub, L. (1986) "The Evolution of Mass Storage", *Byte 11* (5), 161–172.

Ropiequet, S. (1987) CD-ROM *Optical Publishing*, Microsoft Press.

Shermann, C. (1989) *The* CD-ROM *Handbook*, McGraw-Hill.

Ziemer, S. M. (1988) "Design Considerations for CD-ROM Retrieval Software", *Journal of The American Society for Information Science 39* (1), 43–46.

Zoellick, B. (1986) "CD-ROM Software Development", *Byte 11* (5), 177–188.