Optimal Scheduling Algorithms for Tertiary Storage *

Sunil Prabhakar
Department of Computer Sciences
Purdue University
West Lafayette, IN 47907
U.S.A.
sunil@cs.purdue.edu

Divyakant Agrawal Amr El Abbadi
Dept. of Computer Science
University of California
Santa Barbara
CA 93106, U.S.A.
agrawal@cs.ucsb.edu amr@cs.ucsb.edu

Abstract

The ever growing needs of large multimedia systems cannot be met by magnetic disks due to their high cost and low storage density. Consequently, cheaper and denser tertiary storage systems are being integrated into the storage hierarchies of these applications. Although tertiary storage is cheaper, the access latency is very high due to the need to load and unload media on the drives. This high latency and the bursty nature of I/O traffic result in the accumulation of I/O requests for tertiary storage. We study the problem of scheduling these requests to improve performance. In particular we address the issues of scheduling across multiple tapes or disks as opposed to most other studies which consider only one or two media. We focus on algorithms that minimize the number of switches and show through simulation that these result in near-optimal schedules. For single drive libraries an efficient algorithm that produces optimal schedules is developed. For multiple drives the problem is shown to be NP-Complete. Efficient and effective heuristics are presented for both single and multiple drives. The scheduling policies developed achieve significant performance gains over naive policies. The algorithms are simple to implement and are not restrictive. The study encompasses all types of storage libraries handling removable media, such as tapes and optical disks.

Keywords: I/O Scheduling; Optimal, Single-Drive; Multiple Drives; Removable Media; Tape Libraries.

1 Introduction

The ever growing needs of large database systems cannot be met by magnetic disks due to their high cost and low storage density. Examples of such applications include digital libraries, multimedia repositories and video-on-demand. These applications are characterized by very large storage requirements. In the present commercial setting and in the foreseeable future, the only practical solution for storing such large volumes of data is tertiary storage. Although tertiary storage, in particular magnetic tape, has historically been used primarily for archival or backup purposes, the exploding storage requirements and the higher cost of secondary storage are forcing computer architects and designers to re-evaluate the role of tertiary storage[CHL93].

*Work supported by a research grant from NSF/ARPA/NASA IRI9411330, NSF instrumentation grant CDA-9421978, and NSF CAREER grant IIS-9985019.

Commercially available automatic tape and disk libraries or juke boxes (e.g. [Exa96, SON96, Qua96]) provide automated access to large amounts of tertiary storage. These libraries can hold hundreds or thousands of media. Throughout this paper we will use the term *medium* to refer to a tertiary storage unit such as a magnetic tape or an optical or magneto-optical disk. A library typically has a small number of read/write drives and a few robot arms that load and unload media to and from these drives. Upon receiving a request, the robots will load the desired medium onto one of the drives (if there is already a medium loaded on the drive - it may need to be rewound in the case of tapes, ejected and replaced onto the shelf). The drive then loads the new medium, seeks to the requested location and then performs the I/O.

The time required to switch a medium on a drive is usually of the order of a few seconds, and the time required to seek an entire medium may be as high as a few minutes. Typically, for optical disks the seek time is much smaller than the switch time while in the case of tapes the seek time may dominate the switch time. For example, the SONY OSL-2001 juke box can switch optical disks in about 5 seconds and the average seek time for optical disks is 28 - 40 milliseconds [SON96]. The DLT4700 mini tape library has a load/unload cycle of 29 seconds and the average seek time is about 68 seconds [Qua96]. Processors, on the other hand, are many orders of magnitude faster than these switch and seek times. These differences in speed coupled with the fact that I/O traffic is bursty in nature [ABZ96, RW93] increase the potential of a large number of requests accumulating at the library.

Processing requests in first come first serve (FCFS) order in such a system results in poor performance. For example, consider three requests that arrive at the library: the first is for blocks 5 through 10 on tape A, the second is for blocks 1 through 20 on tape B and the third is for blocks 11 through 20 on tape A. If these are processed in FIFO order, we see that tape A will be loaded, then ejected and tape B is loaded which is then ejected and tape A is reloaded and the drive seeks over the first 10 blocks. In contrast, if we service the requests in the order first, third and second, we see a reduction in the total service time since we eliminate one eject, one load and a seek on tape A. Thus, given a set of pending requests at the library, it is desirable to determine an optimal order of execution such that some performance metric is optimized. This metric could be the average waiting time, or the time to completion of the last request or the throughput of the system.

We study the problem of scheduling I/O requests for robotic libraries to improve performance. In particular we study the problem of scheduling requests for multiple media, in contrast to most other studies on scheduling. We develop optimal and near-optimal schedules for robotic storage libraries. Although tapes and optical disks, the two leading tertiary storage technologies today, differ significantly in their access characteristics (one is sequential and the other is random access) our study is general enough to be applicable to both. Preliminary results were presented in an earlier workshop [PAES97].

The rest of this paper is organized as follows. Section 2 describes the problem and discusses related work. Section 3 addresses the problem of minimum switching scheduling on a single drive. The more general problem of scheduling with minimum switching on multiple drives is shown to be NP-Complete in Section 4. In Section 5 we investigate the impact of allowing extra switches. In Section 6 the performance of the optimal and heuristic solutions for the scheduling problems based on simulations are presented. Section 7 concludes the paper.

2 Problem Description

In this section we present the problem of I/O scheduling for robotic libraries and relate it to work that has been done in other areas. It is shown that the problem of robotic library scheduling is different from the problems that have already been studied.

2.1 General Problem

Given an automatic library of removable media (tape or optical disk) consisting of drives, robotic arms and media, and a set of requests, what is the most efficient schedule for the requests? In order to compare the efficiency of potential scheduling policies, it is necessary to define a quantitative measure of performance. This measure, also known as the optimization function, could be one of several possibilities. Some of the popular optimization functions that have been studied in the literature include [Pin95, MP93]: *Makespan:* - the time at which the last request is completed, *TWCT:* - the Total Weighted Completion Time (each job has an assigned weight), and *TWT:* - the Total Waiting Time for all jobs and *Throughput:* - the average number of requests serviced per second. Since the major concern for tertiary storage performance is the high latency, reducing the amount of time that a given request has to wait before it is serviced is important. We have therefore focused on minimizing the average waiting time over all the requests as our optimization function. The waiting time for a request is taken to be the time from the arrival of the request to the beginning of the data transfer for the request. This includes the time to perform necessary seek operations. A more formal definition of the problem is now presented.

Definition 2.1 *Given a set of media, S, a number of drives, d, a set of requests, R, each request $r \in R$ consisting of a medium number, a start block, a number of blocks and a read/write flag, what is the best schedule for servicing these requests such that the average waiting time for all requests is minimized.*

It is assumed that requests do not span more than one medium. Though it is possible that techniques such as striping could be used to split data items across several media, two independent studies [GM95, DK93] have concluded that due to the small number of drives in tertiary libraries, striping is not beneficial in the presence of concurrent users. It is possible that a data item is too large to fit on a single medium, in which case we can split the request into separate requests for each contiguous segment of the data item. Thus the assumption is reasonable.

2.2 Related Work

The problem of task or job scheduling has been extensively studied in the literature [Pin95, MP93]. Related areas are discussed here, with emphasis on the applicability of existing results to our problem. Several variations of the problem of scheduling tasks on processors have been investigated [Par95, GJ79]. The various parameters studied include single versus multiple processors, preemptive versus non preemptive tasks, tasks with and without release times and deadlines, precedence constraints between tasks, tasks with weights or priorities and tasks with setup times. Various optimization functions have also been studied such as *makespan*, *weighted completion time*, *total weighted tardiness* (for tasks with deadlines), *maximum lateness*.

The problem of robotic library scheduling, however, is not addressed by any of the existing studies. We now explore the similarities and underscore the differences between these two. The

drives and processors play analogous roles, as do the requests and tasks. The processing time for tasks however, does not easily map onto any simple concept for the library. It is tempting to relate it to the transfer time, but this does not capture the seek and switch latencies that delay the processing. These latencies could be included as part of the processing time for each request, however this leads to a further complication. This is because the latency of access for a request is not constant – rather it depends largely upon the state of the drives (which medium is currently loaded and where the head is currently located) and the location of the requested data. We therefore have to work with variable task processing times which has not been addressed in earlier studies.

One variation of the processor scheduling problem appears to address a similar problem - tasks with setup times. Problems with sequence dependent setup times have been studied wherein the setup time required for a given request is a function of the previous request. This formulation seems to be most applicable to the problem of robotic library scheduling. However, the studies have focussed on the single processor case and the optimization function used is the *makespan*. In such a situation, the problem of minimizing the makespan reduces to the well known Traveling Salesperson Problem (TSP)[LLKS85]. If, however, the optimizing function considered is the *total waiting time* or the *total completion time*, as in our study, then the library scheduling problem does not reduce to the TSP problem.

The robotic library scheduling problem is also related to the problem of disk scheduling which has been studied extensively in the operating systems community. For disk scheduling, a number of requests need to be serviced on different locations. The drive has to seek between requests that are not contiguously located. Numerous algorithms have been proposed for efficiently servicing disk requests, such as C-SCAN [Teo72] and HEADSCHEDULE [ABZ96]. The major difference between disk scheduling and robotic library scheduling is that tertiary storage media such as tapes or optical disks are removable as opposed to fixed magnetic disks. The algorithms developed for disks assume that all requests are located on the currently loaded medium which is not true for robotic libraries.

More recent work on tape scheduling [HS96, LO96] has studied efficient processing schedules for I/O requests for single tapes. In [HS96] Hillyer and Silberschatz have analyzed algorithms for scheduling batched requests for a single serpentine tape that is loaded on the drive. Li and Orji [LO96] have studied efficient scheduling policies for linear tapes. Both these studies do not consider scheduling requests for more than one medium. Work has also been done on optimizing the performance of relational database management systems that incorporate tertiary storage [SS93, ML95, Sar95a, Sar95b]. Myllymaki and Livny have investigated the benefits of executing tape and disk I/O in parallel [ML96]. In [FM96], a log structured file system for tertiary media has been proposed. The feasibility of striping in tape based systems has been studied by Drapeau and Katz [DK93] and also by Golubchik and Muntz [GM95]. Other studies have looked at the problem of reorganization of data that is stored on tertiary media in order to improve retrieval performance [CDK+95, SS94].

2.3 Problem Specification

It has been shown that I/O traffic is bursty [RW93], i.e. a large number of requests arrive together followed by a long period during which no more requests arrive. Under these settings, the scheduling problem can be simplified by considering the requests in each burst independently. Further, it can be assumed that these requests arrive together, and thus the problem can be viewed as off-line

scheduling with requests arriving together. This technique has also been used for studying disk scheduling policies [ABZ96]. For robotic library scheduling there are two contradictory factors that come into play: reducing seek and switching costs by scheduling requests for the medium that is currently loaded onto a drive versus servicing short seek requests on other media before servicing requests which require large seeks on the currently loaded medium. The importance of the two factors is determined by the relative values of the switch and seek times. If the switch time is large compared to the expected or average seek time, then it is unlikely that switching to a new medium before servicing all requests on the currently loaded medium will be beneficial. On the other hand, if the switch time is small compared to expected or average seek times then switching media without servicing all requests and then later returning to service the remaining requests may result in lower waiting times.

Due to the large magnitude of switching costs, we expect that eliminating switches will be more important. Therefore we begin by analysing schedules that minimize the number of media switches. Since each medium for which there is an I/O request must be loaded onto a drive at least once, it is easy to see that a minimum number of exchanges is achieved by a schedule which loads each medium only once. When a medium is loaded, all requests for it are serviced before it is unloaded. Under these conditions, the scheduling of requests can be viewed as consisting of two independent components: deciding the order in which to load the media and deciding the order in which to service the requests for a loaded medium. The problem of finding efficient schedules for a loaded medium has been studied earlier ([Teo72, ABZ96, LO96, HS96]). Therefore, we do not address this problem. Instead, we assume that one of the existing techniques will be used to service all requests after the medium is loaded. We investigate the problem of deciding the order in which to load the media. After solving the problem of minimum switching schedules, we investigate the implications of allowing extra switches in order to reduce the overall waiting time.

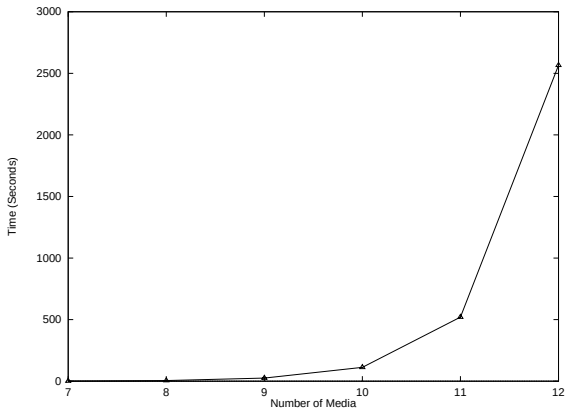


Figure 1: Exhaustive Search Times

Given the current improvements in processor speeds and the high access latency of tertiary storage devices, it is tempting to consider an exhaustiv e search of all possible schedules in order to obtain an optimal solution instead of looking for efficient scheduling policies. This approach is feasible only for very small numbers of media. In Figure 1 we present the times required to conduct exhaustive searches for different numbers of media. The graph clearly shows an exponential increase in the time required. Note that we have only considered permutations of the media and not the

requests. Given that the number of requests is generally much larger than the number of media, if all permutations of the requests are considered, then exhaustive searching will be even slower. For even as few as 12 media, we require almost half an hour to compare all the possibilities.

3 Minimum Switching Scheduling on a Single Drive

We begin by studying the problem of *Minimum Switching* scheduling for a single drive. As discussed earlier, the problem simplifies to determining the order in which the media are loaded onto the drive. Once a given medium is loaded, all requests for that medium are serviced before it is ejected from the drive. The order of processing the requests optimally within a medium is not important for our discussion. This order can be as simple as the order of the block numbers for linear tapes or more complex such as those suggested by Hillyer and Silberschatz [HS96] for serpentine tapes or any disk scheduling policy for optical disks. When all requests have been serviced, the medium is rewound to the beginning (in the case of tapes) and ejected.

3.1 Optimization Function

In order to understand the factors involved in the scheduling problem, we derive an expression for the average waiting time. All requests are assumed to arrive at the same time. The average waiting time is simply the total waiting time for all requests divided by the number of requests. Consequently, for any given set or requests, minimizing the average waiting time is the same as minimizing the total waiting time. In the remainder of the paper we use the total waiting time instead of the average waiting time since both reflect the same optimization function. Since all requests arrive together (at time 0), we define the optimization function as follows:

Definition 3.1 *The total waiting time for a schedule of n requests is the sum of the waiting time of all requests. The waiting time for a request is the time (since arrival) at which the data transfer for the request is begun.*

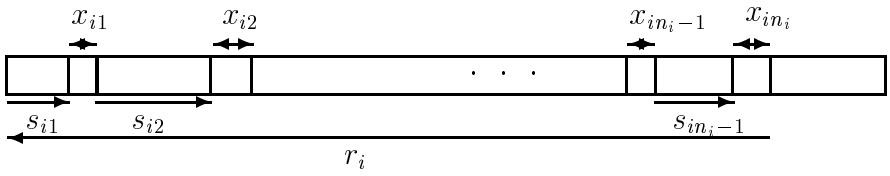


Figure 2: Notation used in the development for Tapes

The goal of our scheduling policies will therefore be to minimize the value of the total waiting time. The ordering of the media refers to the sequence in which they are loaded onto the drive. Although tapes and disks differ significantly, our study is general enough to be applicable to both. In order to see how this is achieved, consider Figure 2 and Figure 3. The servicing of requests on both types of media can be viewed as beginning with the head positioned at some starting location followed by an alternating sequence of search (seek and rotational delays - s_{ij}) and transfer (x_{ij}) operations. Finally there is a rewind operation (r_i) which can be significant for tapes but not for disks. Therefore conceptually the processing of requests on the two types of media is the

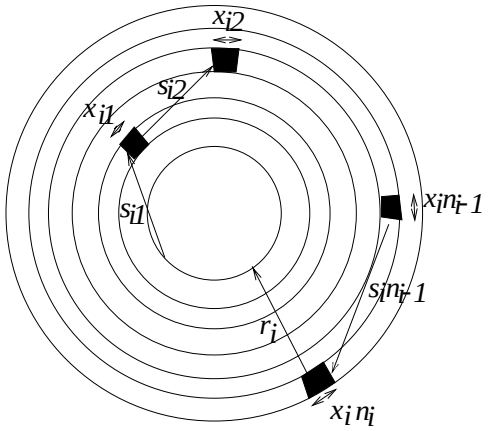


Figure 3: Notation used in the development for Optical Disks

same, although the magnitudes of the various times will be different. In both figures the order of processing of requests, $1, 2, \cdots, n-1, n$ is determined by the scheduling algorithm used for requests within a single medium. An expression for the total waiting time is given by Equation 1. The details of its derivation are presented in Appendix A.

$$W = \sum_{i=1}^{m} \left(C_i + T \cdot i \cdot n_i + n_i \sum_{k=1}^{i-1} P_k \right) \tag{1}$$

The total waiting time for all requests can be viewed as consisting of three components. The first component, C_i, is the waiting time incurred by requests for medium i after the medium has been loaded onto the drive. This time is independent of the order in which the media are loaded. The second component, $T \cdot i \cdot n_i$, is the waiting time for requests of medium i due to the switching of media that precede medium i. This time is the same for all requests for medium i and is dependent only on i. The third component, $n_i \sum_{k=1}^{i-1} P_k$, is the waiting time for requests for medium i due to the processing of the requests for the preceding media. It is the same for each request of medium i and is equal to the sum of the processing times for the preceding media. Breaking up the total waiting time in this manner is instrumental in determining an optimal solution to the single drive problem.

3.2 Optimal Solution

We will now develop an algorithm that determines optimal schedules for single drive libraries. For this purpose we need to establish the following property and lemma.

Property 1 *In any given schedule, exchanging any two media in the order, affects the waiting times for requests on the media that are exchanged and all media that lie between them only. The waiting times for requests for media that precede or follow both exchanged media remain unchanged.*

The above property is a direct consequence of the impact of the three components in the waiting time when the media are reordered.

Lemma 3.1 *In any optimal schedule, $O_1, O_2, \cdots, O_m$, the following condition must hold:*

$$\forall i \;\; 1 \leq i < m, \;\; \frac{n_i}{T + P_i} \geq \frac{n_{i+1}}{T + P_{i+1}} \tag{2}$$

The proof of Lemma 3.1 is presented in Appendix B. We now develop a sufficient condition for an optimal solution.

Theorem 3.1 *Any schedule of m media which satisfies Equation 2 is optimal.*

Proof: Suppose that there is a schedule, $A_1, A_2, \cdots, A_m$ which satisfies Equation 2, but is not optimal. Let $O_1, O_2, \cdots, O_m$ be an optimal schedule. If both schedules are the same, then both are optimal, otherwise, let i be the smallest index for which the two schedules differ. That is $\forall j, \;\; 1 \leq j < i, A_j = O_j$, and $A_i \neq O_i$. Since both schedules satisfy Equation 2, $\frac{n_{O_i}}{T+P_{O_i}} \geq \frac{n_{O_k}}{T+P_{O_k}}$, $\forall k, \; i < k \leq m$, and $\frac{n_{A_i}}{T+P_{A_i}} \geq \frac{n_{A_k}}{T+P_{A_k}}$, $\forall k, \; i < k \leq m$. In other words, A_i must have the largest ratio of number of requests to processing time plus switch time for all media that follow A_i, and O_i must also have the largest ratio for all media that follow O_i. This can only be true if either $A_i = O_i$ or the ratio for A_i is equal to the ratio for O_i. Thus we observe that the only difference in the two sequences can be the order in which media that have the same ratio are ordered. Next we show that rearranging the order of media that have the same ratio has no effect on the overall waiting time.

Consider a sequence of media, $C_l, C_{l+1}, \cdots, C_n$, all of which have the same ratio of number of requests to processing time plus switch time. If we exchange any two of these media, say, i and j (C_i precedes C_j in the above sequence), then the change in the total waiting time is:

$$\begin{aligned}
\Delta W &= \sum_{k=i}^{j} (\textit{change in waiting time for medium } k) \\
\Delta W &= \Delta W_j + \Delta W_{j-1} + \cdots + \Delta W_{i+1} + \Delta W_i \\
\Delta W_j &= n_j(-P_i - T - P_{i+1} - T \cdots - P_{j-1} - T) \\
\Delta W_{i+1} &= n_{i+1}(P_j + T - P_i - T) = n_{i+1}(P_j - P_i) \\
&\vdots \\
\Delta W_i &= n_i(P_{i+1} + T + \cdots + P_j + T)
\end{aligned}$$

Therefore,

$$\Delta W = n_i(P_{i+1} + T) - n_{i+1}(P_i + T) + \cdots + n_i(P_j + T) - n_j(P_i + T)$$

Since all these media have the same ratio, it is clear that the change $\Delta W = 0$. Hence we can interchange any pair of media in the sequence without affecting the total waiting time. In other words, any permutation of media with the same ratio has the same waiting time.

Since the two sequences $A_1, A_2, \cdots, A_m$ and $O_1, O_2, \cdots, O_m$ differ only in the sequence of media with the same ratios, they must have the same total waiting times. In other words the sequence $A_1, A_2, \cdots, A_m$ is also optimal. □

Finding an optimal minimum switching schedule on a single drive is therefore reduced to the problem of sorting the media in non-increasing order of the ratio of number of requests to the processing time plus switch time. This is easily accomplished by using any efficient sorting algorithm. The time complexity for determining the optimal order is $O(N + m \log m)$, where N is the

number of requests and m is the number of media with requests. The $O(N)$ component is the time required to estimate the processing time, P_i, for each medium. This time is determined by the scheduling algorithm used within a medium. (We are assuming that the time to estimate the total processing time depends linearly upon the number of requests). Note also that the order of processing of requests within a medium is not restricted, i.e. any of the existing scheduling schemes (e.g. [HS96, LO96, ABZ96]) can be used. It is necessary only to be able to determine the seek and transfer times for each request. We have not made any assumptions about the seek function or transfer rates for the media either (they can be arbitrarily distributed).

4 Multiple Drives

We now consider the scheduling problem in the presence of more than one drive. For the multiple drive case, the total waiting time can be viewed as the sum of the waiting times for each of the drives. In Appendix C, an expression for the total waiting time for all requests, assuming d drives, is derived. The total waiting time can be written as:

$$W = K + \sum_{k=1}^{d} \sum_{i=1}^{m^k} \left(n_i^k \sum_{j=1}^{i-1} (P_i^k + T) \right) \tag{3}$$

where $K = \sum_{k=1}^{d} \sum_{i=1}^{m^k} \left(C_i^k + T \cdot n_i^k \right)$ is a constant independent of the assignment of media to drives or the order of processing of media on drives. The superscripts identify the different drives, i.e., m^k refers to the number of media to be scheduled on drive k, n_i^k and P_i^k refer to the number and processing time of all requests for the ith medium to be scheduled on drive k respectively.

We now show that the problem of finding a scheduling policy which minimizes Equation 3 is the same as a known NP-Complete problem. Let us first note that the problem of minimizing the total waiting time is the same as the problem of minimizing the total completion time. The only difference between the two measures is a constant amount, equal to the sum of the transfer times for all the requests. Since this amount is independent of the order of processing, we can say that the total completion time for all requests is given by

$$W = K' + \sum_{k=1}^{d} \sum_{i=1}^{m^k} \left(n_i^k \sum_{j=1}^{i-1} (P_i^k + T) \right) \tag{4}$$

where $K' = \sum_{k=1}^{d} \sum_{i=1}^{m^k} \left(C_i^k + T \cdot n_i^k + \sum_{j=1}^{n_i^k} x_{ij}^k \right)$ is also a constant independent of the schedule. Since NP-Complete problems are usually posed as a Yes or No answer, we restate the problem as:

Definition 4.1 Multidrive Minimum Switching Scheduling (MMSS)
GIVEN: *A set of media (S, $|S| = m$), number of drives (d), a medium switching time (T), number of requests (n_i) and processing time (P_i) for each medium $i \in S$, and a positive integer J.*
QUESTION: *Is there a d-drive schedule for the media such that the value of W defined by Equation 4 is no more than J?*

Theorem 4.1 *The problem of multidrive minimum switching scheduling is NP-Complete.*

The proof of this theorem is presented in Appendix D. Since no efficient solutions for NP-Complete problems are known, we propose two heuristics that give very good performance. In particular, the OPT algorithm which gives optimal schedules for single drive libraries and an algorithm that schedules media based solely upon the number of requests are proposed as heuristics. As seen in Section 6, both these heuristics give very good performance for the multiple drive case.

5 Impact of Extra Media Switches

In the last section we studied schedules that minimize media switches. In this section we investigate whether minimizing switches is always beneficial. First the complexity of the problem is demonstrated through analysis of the result of introducing an extra switch. Then based upon simulation we show that extra switches are rarely beneficial and the gains are small.

5.1 Analysis of Extra Switching

We expect that for certain media such as optical disks which have very small seek times compared to switch times, it is likely to be beneficial. On the other hand, for media such as tapes, for which the switch time can be smaller than the seek or transfer times, the answer is not obvious. Consider the following example.

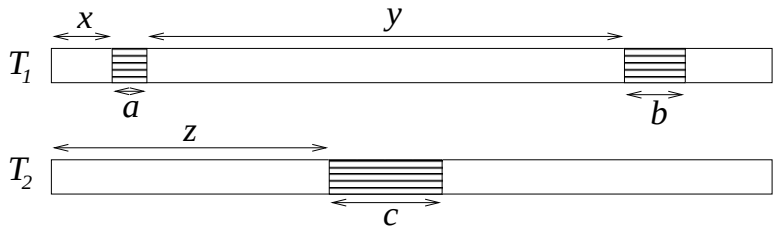


Figure 4: Layout of Requests on Tapes (Not drawn to Scale)

Example Consider two linear tapes, T_1 and T_2, with three requests of lengths a, b and c bytes as shown in Figure 4. The separation of the requests is given by x, y and z in bytes. To minimize swithces each tape is loaded once and its requests are serviced. The optimal order for loading the two tapes is given by the algorithm OPT described in Section 3, i.e. load tapes in non-increasing order of the ratio of number of requests to the sum of the switch time and the processing time. Let us assume that the ratio for T_1 is higher than that for T_2, i.e.,

$$\frac{n_1}{T + P_1} \geq \frac{n_2}{T + P_2} \tag{5}$$

where P_1 and P_2 are the processing times for the two tapes and T is the switch time. Request c waits for seeking x bytes, reading a bytes, seeking y bytes, reading b, rewinding $b + y + a + x$ bytes, a switch and seeking z bytes. If y is very large, then the time to seek and rewind over y can be much larger than the time to switch tapes. Therefore we could process a, then switch tapes, process c and then switch tapes and process b and have a lower average waiting time. The size of y is limited by the length of the tape and also the requirement of equation 5. must be large. If we assume that $s = r$ then y must be at least as large

Motivated by the above example which shows us that introducing extra media switches can be beneficial, we now analyze the conditions under which it will be beneficial in general. We will focus on a single drive. Given a set of requests, the OPT algorithm is used to determine an optimal sequence with minimum switching as described in the previous section. The schedule for the requests on a single medium after the medium is loaded is left unspecified. As mentioned earlier, any of the existing schedules can be used. We require to know only the sequence in which the requests will be serviced and the total processing time.

Let us now consider the benefit of offloading the ith medium after the jth request for the medium has been serviced. The remaining requests for this medium will be serviced at a later time when the medium is reloaded. Other media could be loaded and serviced completely before the offloaded medium is reloaded. Any of the m media could be split in this manner. Figure 5(a) shows the optimal sequence of the media before the split. Let M be the medium that we choose to offload and reload. In general there will be some set of media A that precede M and some set B that follow M in the optimal minimum switching schedule generated by OPT. The offloading and reloading of M can be viewed as the replacement of medium M by two media, M_1 and M_2, where M_1 contains the data for the first j requests of M and M_2 contains the data for the rest of the requests. The locations of the requested data on M_1 and M_2 are the same as the locations of the respective data on M.

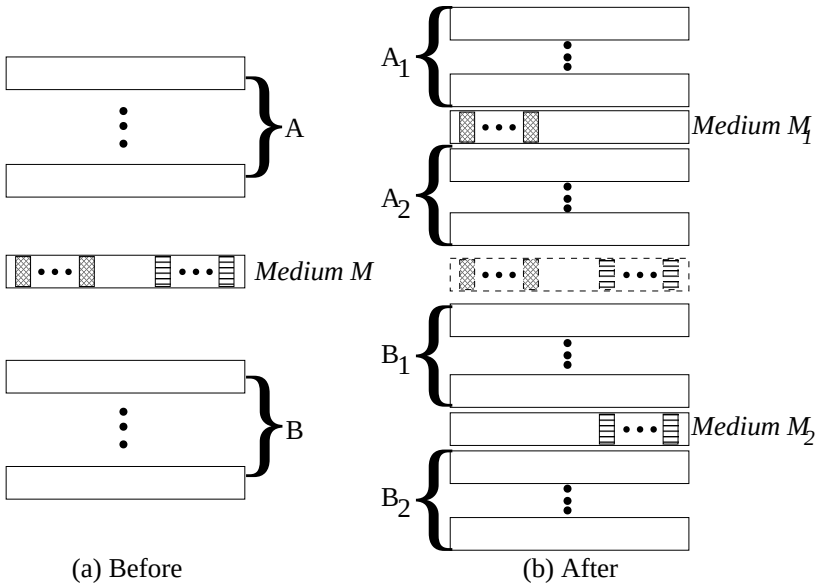


Figure 5: Optimal sequence of Media (a) Before Split (b) After Split

Having split medium M into two media, we can now look for the optimal sequence of this new set of media using the OPT algorithm. This new order would give us the best schedule for the media if we allowed medium M to be offloaded after its jth request. Since the optimal order is based upon the ratio for each medium, the media in the sets A and B will remain in the same relative order as before since their ratios are unchanged. Media M_1 and M_2 will be inserted into the schedule depending upon their ratios. Due to the large variability in the relative values of the various tertiary technology parameters such as, switch time, seek and rewind rates and overheads, transfer rates, it is possible that the ratios of the three media M, M_1 and M_2 could have any of

the six possible orderings.

For example, M_1 could have the highest ratio, followed by M, and M_2 could have the lowest ratio, or M_2 could have the highest ratio and M could have the lowest. In order to demonstrate the complexity of the factors involved, we will analyze only a single representative case. Since we are interested in introducing extra switches only for the purpose of reducing the average waiting time, we choose to analyze the case that we expect to be most likely to result in reduction of total waiting time. This case is shown in Figure 5(b). The order of non-increasing ratios for the three media is M_1, M and M_2. Since M_1 has a higher ratio than M, it is possible that it moves ahead of some of the media in set A. Thus set A is now split into two sets, A_1 which is scheduled before M_1, and A_2 which is scheduled after M_1. Similarly, M_2 has a ratio that is lower than M, therefore it is possible that it gets scheduled after some of the media in set B, splitting it into B_1 and B_2.

It should be noted that any of the four sets of media A_1, A_2, B_1 and B_2 could be empty as long as $A = A_1 \bigcup A_2$ and $B = B_1 \bigcup B_2$. Note that medium M, shown dotted in Figure 5(b), is not present in that schedule – it is only shown to indicate the division between A_2 and B_1. Also, M_1 and M_2 are the same medium (M) being loaded at different times in the same schedule, they are not independent media.

Let us now consider the effect of this split of medium M on the total waiting time. The requests on the media in set A_1 will see no change in their waiting times. Each request on the media in set A_2 will experience an increase in waiting time of an extra switch and the processing time of medium M_1, i.e. $T + P_{M_1}$. The change in waiting time for each request can be analyzed similarly. More formally, using n_x to denote the number of request in set x, P_x to denote the total processing time required for all requests in set x, N_x to denote the number of media in set x and ΔW_x to denote the total change in the waiting time for all requests in set x:

$$\begin{aligned}
\Delta W_{M_1} &= -n_{M_1}(N_{A_2}T + P_{A_2}) \\
\Delta W_{A_2} &= n_{A_2}(T + P_{M_1}) \\
\Delta W_{B_1} &= n_{B_1}(P_{M_1} - P_M) \\
\Delta W_{M_2} &= n_{M_2}(N_{B_1}T + P_{B_1} + T + P_{M_1} + X_2 - X) \\
\Delta W_{B_2} &= n_{B_2}(T + P_{M_1} + P_{M_2} - P_M)
\end{aligned}$$

where X_2 is the time required to seek from the start of M_2 to the first request on M_2 and X is the time required to seek from the end of the jth request on M to the beginning of the $j+1$th request.

In order for the extra switch to be beneficial, the total change in waiting time should be negative. Thus from the above equations, we get:

$$\begin{aligned}
&T(n_{A_2} + n_{B_2} + n_{M_2}N_{B_1} + n_{M_2}) + P_{M_1}(n_{A_2} + n_{B_1} + n_{B_2} + n_{M_2}) + P_{M_2}n_{B_2} + n_{M_2}P_{B_1} + n_{M_2}X_2 \\
&\qquad < \; n_{M_1}N_{A_2}T + P_M(n_{B_1} + n_{B_2}) + n_{M_1}P_{A_2} + n_{M_2}X \qquad (6)
\end{aligned}$$

Also, because of the order in which the media are scheduled by OPT, there are limitations on the value of their ratios. In particular:

$$\left.\frac{n_i}{T + P_i}\right|_{i \in A_1} \geq \frac{n_{M_1}}{T + P_{M_1}} \geq \left.\frac{n_i}{T + P_i}\right|_{i \in A_2} \geq \frac{n_M}{T + P_M} \geq \left.\frac{n_i}{T + P_i}\right|_{i \in B_1} \geq \frac{n_{M_2}}{T + P_{M_2}} \geq \left.\frac{n_i}{T + P_i}\right|_{i \in B_2} \qquad (7)$$

For the split to be beneficial, Equations 6 and 7 need to be satisfied. Equation 7 sets limits on the relative vales of the parameters and Equation 6 requires that:

- n_{M_1} should be large, P_{M_1} should be small,
- N_{A_2} should be large, n_{A_2} should be small, P_{A_2} should be large,
- N_{B_1} should be small, n_{B_1} should probably be large, P_{B_1} should be small,
- n_{M_2} should probably be small, P_{M_2} should be small, and
- n_{B_2} should be small

Some of the factors that determine whether an extra switch will be beneficial depend upon requests for other media, which makes it difficult to determine when an extra switch will be effective. Therefore, it is not obvious when the processing of the next request for the loaded medium should be postponed and requests of other media be processed. Similar expressions can also be derived for the other possible orderings of M, M_1 and M_2. Due to the large number of variables, the analysis is complex and we expect that simple heuristics will not be easy to find. Therefore, we approach the problem through simulation. In the next section, we investigate experimentally the utility of extra media switches.

5.2 Experimental Results

We first present the simulation setup used to conduct the experiments and then the results of the experiments.

5.2.1 Experimental Setup

The experiments were conducted using an event driven simulator for robotic libraries developed using the CSIM [Sch86] simulation package. A model of a general robotic library consisting of a configurable number of drives, robot arms and media has been developed. The performance of the drives and robot arms and the characteristics of the media are also configurable. Table 1 lists some of the parameters that can be adjusted for the model. The model mimics an actual robotic library. The drives operate independently of each other, except that they share the robot arms to pick and put media.

Experiments were conducted using a single drive setting only. Two sets of input request patterns were used for our experiments. The first set consisted of requests distributed uniformly across the media in the library. The second set consisted of a skewed "hot-cold" distribution where 80% of the media were accessed 20% of the time and the remaining 20% of the media were accessed 80% of the time. For each experiment, the number of media was varied from 10 to 100, and the number of requests was always set to be ten times the number of media, e.g. for 20 media, we generated 200 requests. Each experiment was repeated thrice using different random number seeds for the request generator. Due to the lack of space, not all results are presented, only representative results are shown and discussed in the paper.

5.2.2 Validation of Hypothesis

Based upon the large value of the exchange time as compared to the average seek and transfer times for optical and magneto-optical disks, we expect that introducing extra switches will not be

Parameter	*Meaning*
NDRIVES	Number of drives
NROBOTS	Number of robotic arms
NUM_MEDIA	Number of media in the library
NUM_REQ	Number of requests for each experiment
PICK_TIME	Time required by robot to remove medium from drive
PUT_TIME	Time required by robot to place medium on drive
MOVE_TIME	Time required by robot to move between media on shelf
SEEK_OVHD (REW_OVHD)	Constant overhead for seek(rewind)
SEEK_RATE (REW_RATE)	Rate at which the drive can seek(rewind)
XFER_SPEED	Transfer rate of the drive
REQ_SIZE	Size of a single request
EJECT_TIME	Time required by the drive to eject a medium
LOAD_TIME	Time required by the drive to load a medium

Table 1: Table of Model Parameters

beneficial. Several experiments were conducted for a typical optical disk juke box (SONY OSL-2001 with SMO F541 drives). The values of the various parameters used are given in Table 2. As expected, in all our experiments we found that introducing an extra switch was always harmful and resulted in increases average waiting. The percentage increase varied depending upon the number of disks and requests, being greater for smaller numbers. The increase was greater for the hot-cold distributions than for the random distributions. We found that for 10 disks with the hot-cold distribution, a single extra switch could result in as much as 23% increase in the average waiting time for all requests. Figure 6(a) shows the results for the case of 20 disks with a hot-cold distribution and Figure 6(b) shows the results for the case of 20 disks with a uniform distribution of requests. In each experiment the effect of introducing a switch after each request of each medium (except the last request to be processed on the medium) was evaluated. The x-axis of the plots shows the request number following which the switch was introduced. The requests were totally ordered by ordering the disks. For each disk the order was chosen to be the order of block numbers. The y-axis shows the percentage increase in the total waiting time as a result of the extra switch. Each peak in the plots represents the requests for one disk. Thus for the optical disk, introducing extra switches is not beneficial. The schedules developed by the OPT algorithm for optical disks are therefore optimal.

For tapes however, introducing switches did result in improvements over the minimum switching schedules. Figure 7 shows the results for experiments based upon the data for an Exabyte EXB-480 library and EXB Mammoth drive. The values of the parameters used are given in Table 2. Figure 7(a) shows the results of 20 tapes with a hot-cold distribution and Figure 7(b) shows the results for 20 tapes with uniform distribution. Although gains are observed, they are very small (less than 0.6%) and occur only for a few cases. A natural question to ask is how one could identify offload points that would result in reduction of the total waiting time.

As pointed out earlier, we expect that reductions will be seen when the gap between the location of the request after which the switch is made and the location of the next request on the same tape

Parameter	*OPTICAL DISKS*	*TAPES*
NDRIVES	1	1 or 4
NROBOTS	1	1
NUM_MEDIA	10 - 100	10 - 100
NUM_REQ	100 - 1000	100 - 1000
PICK_TIME	1 sec	10 sec
PUT_TIME	1 sec	10 sec
MOVE_TIME	1 sec	2 sec
SEEK_OVHD (REW_OVHD)	0.0083 (0.0083) sec	0.1 (0.1) sec
SEEK_RATE (REW_RATE)	103 (103) MB/s	193 (188) MB/s
XFER_SPEED	2.0 MB/s	3.0 MB/s
REQ_SIZE	100 KBytes	2560 KBytes
EJECT_TIME	1 sec	8 sec
LOAD_TIME	2 sec	10 sec

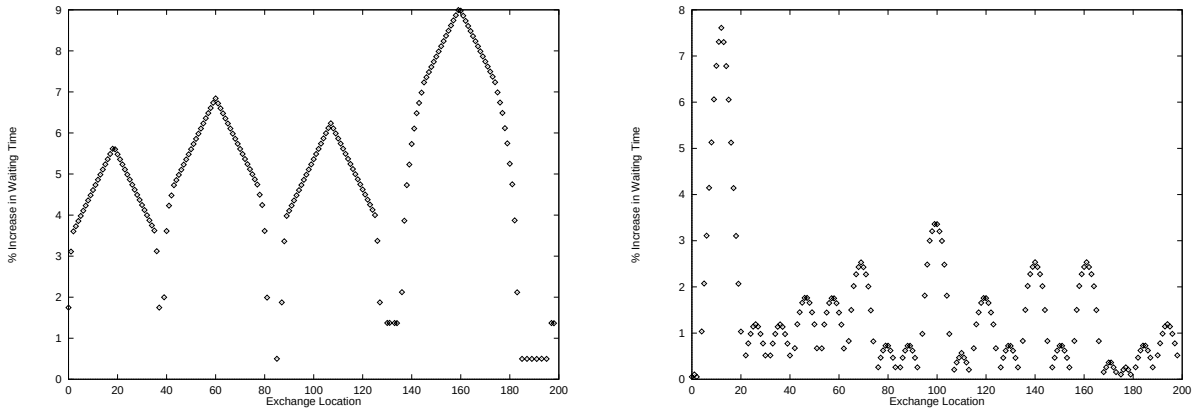
Table 2: Parameter values used for the experiments

is large. To test this hypothesis, in Figure 8, we plot the percentage increase in waiting time versus the distance (in blocks) between the switch location and the next request. The performance with the hot-cold distribution fits the hypothesis, whereas the performance with the random distribution does not.

With the hot-cold distribution, switching is not beneficial when the gap is small, but as the gap gets larger, switching begins to be useful. This behavior was observed for all experiments with the hot-cold distribution. With the random distribution however, there are instances where switching with even a small gap results in reduction of the waiting time. This is due to the influence of the other media in the schedule. With a hot-cold distribution, most of the requests are located on a few tapes and there are tapes with very few requests. Since the dominant factor in the ratio used by OPT is the number of requests, we see that the schedules have the hot tapes loaded before the cold ones. For the random distribution, the number of requests per tape is more uniform and therefore even the tapes at the tail end of the schedule have large numbers of requests.

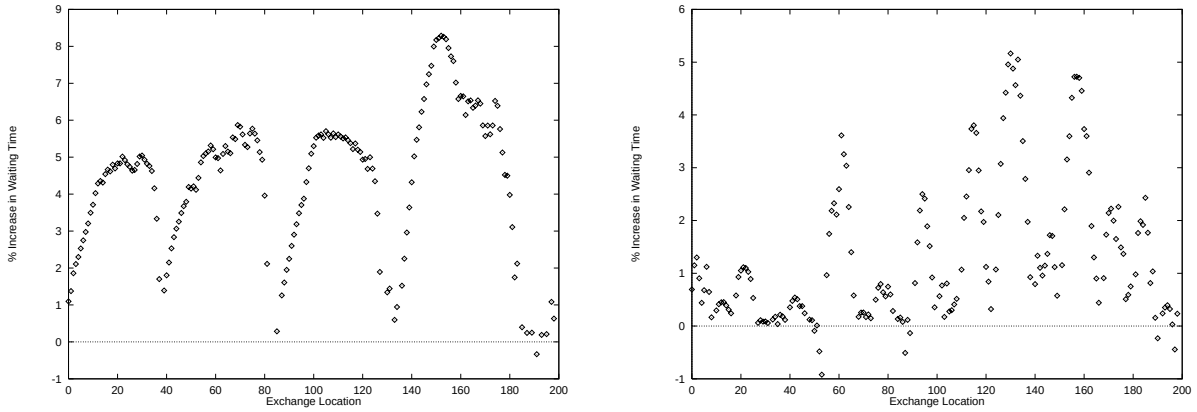
Using the notation of Subsection 5.1, we see that the location of M_2 in the schedule after splitting will be determined by its ratio, which will also be largely governed by the number of requests. With the random distribution, the number of requests in M_2 will usually be smaller than the number of requests in the rest of the tapes. Consequently, it will often be scheduled last, i.e. the set B_2 will be empty. With a hot-cold distribution, the set B_2 will usually be non-empty. This factor makes it more likely that extra switches will be more beneficial with the random distribution than for the hot-cold distribution.

It should be noted that reduction in the total waiting time occurs rarely and when it does occur, the magnitude of the improvement is not large. On the other hand, most of the extra switches lead to increases in the total waiting time, the magnitude of which can be very large (up to 23%). By eliminating unnecessary switches, the OPT algorithm was able to achieve significant improvements over first-come-first-serve scheduling, whereas, the gains of introducing extra switches are not significant. In fact, for some technologies, such as disks, there are no gains. Due to the



(a) Hot-Cold (b) Random

Figure 6: Plot for 20 Disks



(a) Hot-Cold (b) Random

Figure 7: Plot for 20 Tapes

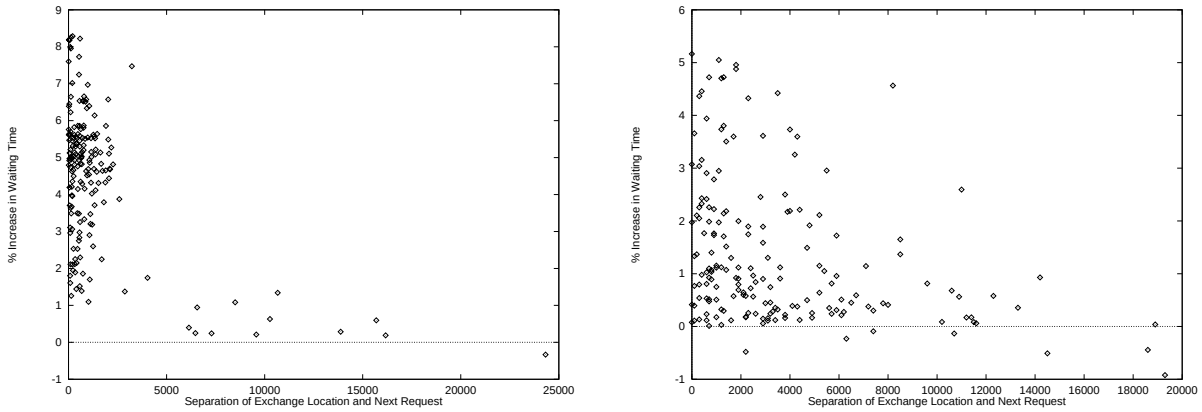
many factors involved it is not easy to determine a criterion that determines when an extra switch should be introduced.

6 Performance Simulation and Heuristics

In this section we present simulation results to validate the performance of the scheduling algorithms discussed in the earlier sections. The evaluation is based upon simulations using the simulator described in the last section. First the benefits of *Minimum Switching* scheduling are presented. Next we propose heuristics for both single and multiple drives and evaluate their performance.

6.1 Validation of *Minimum Switching* Scheduling

In order to validate our hypothesis that *Minimum Switching* scheduling is beneficial, we conducted experiments to compare the performance of variants of the naive First Come First Serve (FCFS) policy. Three versions of FCFS algorithms were examined.



(a) Hot-Cold (b) Random

Figure 8: Increase in Waiting time versus gap for 20 Tapes

The first, called FCFS, schedules requests in the order in which they are received. Note that even though we assume that the arrival time for all requests is the same for the purposes of our waiting time derivation, we do have the notion of ordered arrival. The reason for this is that the order of magnitude of the average waiting time for tertiary storage devices is a few tens or hundreds of seconds, whereas the requests will arrive within a few milliseconds of each other due to an I/O burst.

The second variation, called FCFS_II, is a minimum switching scheduling algorithm which determines the order of media according to the order of the requests. That is, the first medium is the one on which the first request is located, the second medium is the one on which the earliest request not for the first medium is located, and so on. FCFS_II serves all requests on a given medium before ejecting the medium, however it serves these requests in the order in which they arrive.

The third algorithm, called FCFS_III, optimizes over FCFS_II by processing the requests on a given medium in a more efficient manner (such as in the order of ascending tape blocks) depending upon the best algorithm for the media type (tape or disk etc.). The results for single drive and multiple drives were very similar and therefore only the results for the multiple drive setting are shown. Figure 9 shows the performance of the three FCFS policies with both random and hot-cold request patterns. It can be observed that the algorithms that service all requests for the loaded medium before those for other media (FCFS_II and FCFS_III) reduce the average waiting time for requests significantly. Reordering the requests for a given medium results in even further reductions in the average waiting time (FCFS_II versus FCFS_III). This confirms our hypothesis that minimum switching scheduling is beneficial even though it may not be optimal.

It is important to note that these experiments were based upon data for tape libraries for which the seek time dominates the switch time. Hence, we expect that minimum switching scheduling will result in even greater gains when the switch time dominates the seek time such as for optical disk libraries.

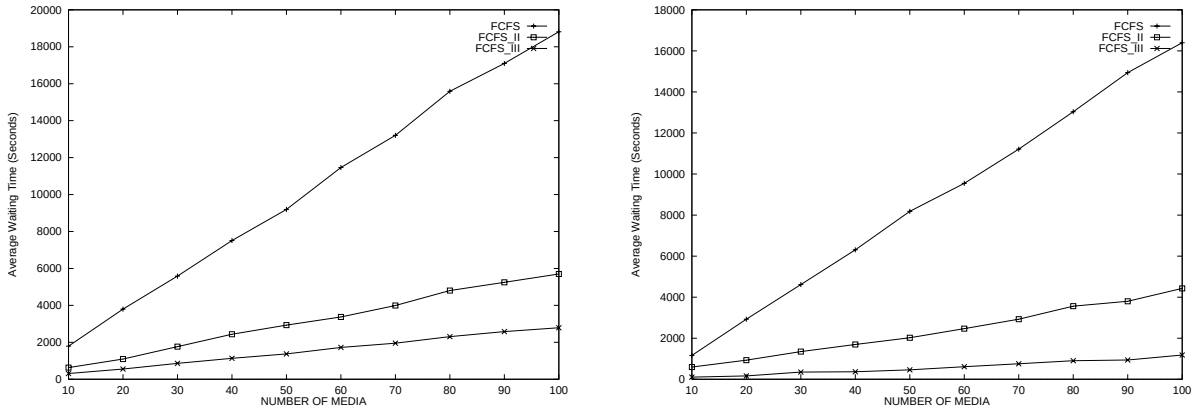


Figure 9: Variations of FCFS for multiple drives with (a) Random and (b) Hot-Cold requests

6.2 Performance of Heuristics

For a single drive, we have shown that an optimal algorithm exists. The algorithm, which we call OPT, schedules media in non-increasing order of the ratio of number of requests to the sum of the processing time and the switch time. Although the optimal algorithm for single drives, OPT, is simple, it requires knowledge of the processing time for each medium. It may not always be possible to determine this time. For example, the scheduling algorithm for requests on a given medium may not be known or the exact layout of the data on the medium may not be known.

In the absence of such information, we propose the following heuristic: *Schedule the media in the order of non-increasing number of requests.* This heuristic, which we call *Number*, does not guarantee optimal solutions but is very easy to implement as it requires no knowledge of the processing or switch times. Several experiments were conducted to evaluate the effectiveness of this heuristic. The results are shown in Figure 10 for Random and Hot-Cold distributions respectively. As can be seen, *Number* performs almost as well as OPT. In particular, it is within 1% of the optimal for all the experiments. It should also be noted that OPT and *Number* perform better than FCFS_III which does not reorder the media.

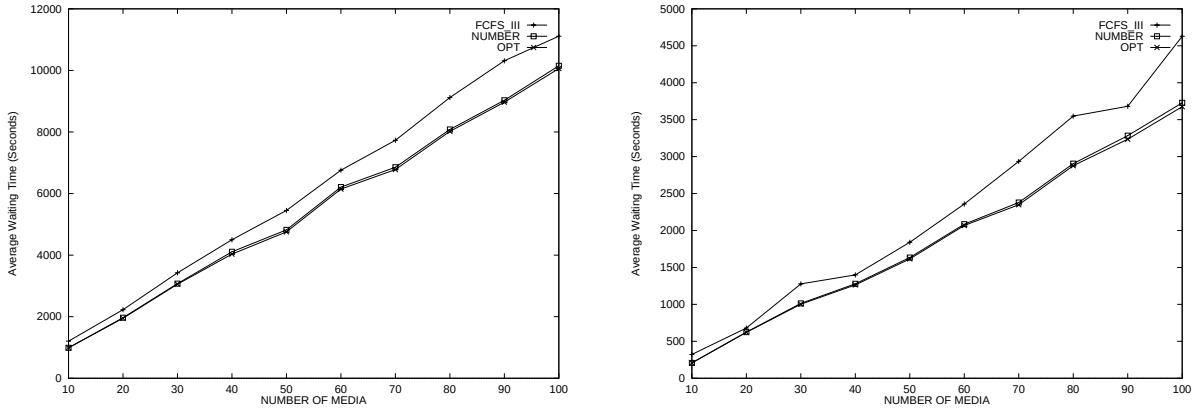


Figure 10: Comparison of the *Number* heuristic with OPT for single drives with (a) Random and (b) Hot-Cold requests

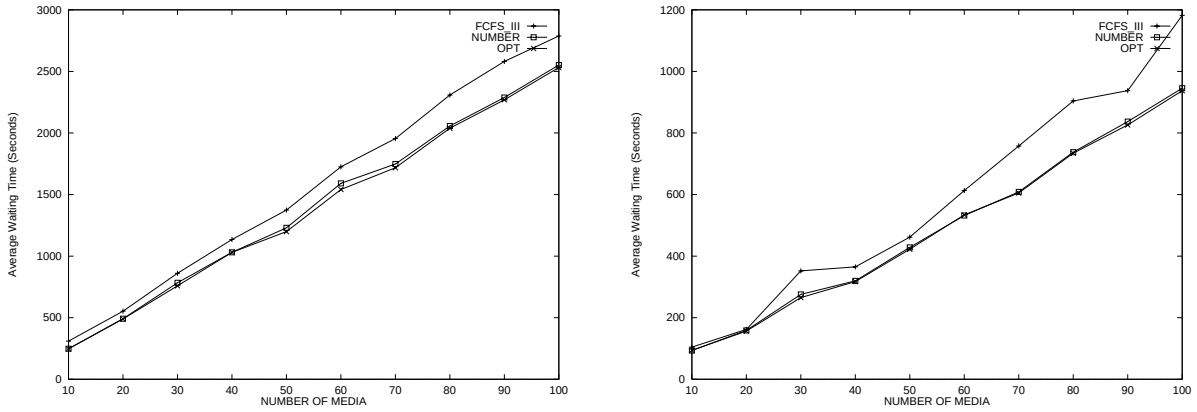


Figure 11: Comparison of the *Number* heuristic with OPT for multiple drives with (a) Random and (b) Hot-Cold requests

For multiple drive scheduling the NP-Completeness result implies that no efficient optimal solution may exist. We therefore look for approximate solutions that give reasonable performance. The simple algorithm, OPT, that generates an optimal solution for the single drive case (order tapes in non-increasing order of the ratio of number of requests to the sum of the processing time and switch time) does not guarantee optimal results for the multidrive case. The analogous algorithm for the minimum weighted completion time problem ($MWCT$)[LKB77] has been shown to have an upper bound. In particular the algorithm is guaranteed to produce a solution that is within $(1 + \sqrt{2})/2$ (approximately within 20%) of the optimal for $MWCT$ [Pin95]. This bound does not necessarily hold for the multiple drive problem due to the other constant factors in the cost. Given its good performance guarantee for the $MWCT$ problem, we evaluated this algorithm for the multiple drive problem. We also evaluated the simpler heuristic based solely on the number of requests per medium, *Number*.

Since no efficient optimal algorithm was developed for the multiple drive case, we used a brute force algorithm to generate the lower bound for the purpose of comparison. Given a set of m media and d drives, there are d^m different allocations of media to drives. Given a set of media to be processed on a drive, the lower bound schedule of these media can be determined using OPT, the optimal algorithm for single drives. Therefore, for each allocation of media to drives, we can determine an optimal schedule. The lower bound for the multiple drives is taken to be the allocation with the least average waiting time. Since determining the lower bound requires exponential running time, as explained in Section 2 , we were limited to small numbers of media. Several experiments were conducted with 10 media using both request distributions to compare the performance of OPT and the lower bound.

The results of these experiments are shown in Figure 12. It can be seen that the OPT algorithm for single drives performs surprisingly well in the multiple drive setting as well. In most cases, OPT produces schedules that are within 1% of the lower bound for multiple drives. The performance of *Number* was compared with that of OPT for larger numbers of tapes with multiple drives. The results which are very similar to those for a single drive are shown in Figure 11. It can be seen that the *Number* heuristic generates solutions that are very close to those produced by OPT. As for the single drive case, both OPT and *Number* perform better that FCFS_III with multiple drives. For example, the average waiting time for FCFS with 80 media and the random distribution is 14492

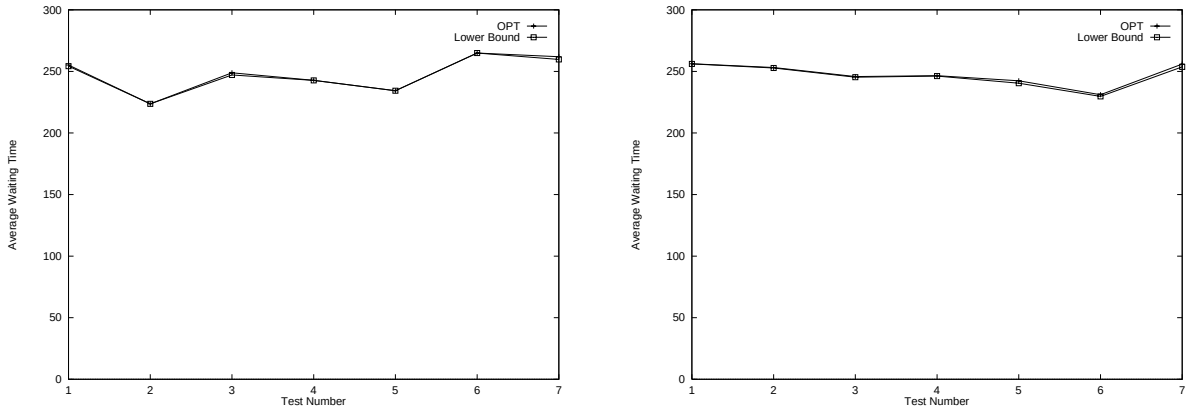


Figure 12: Comparison of OPT with the lower bound for multiple drives with (a) Random and (b) Hot-Cold request distributions

(Figure 9) seconds whereas the time for OPT is 1952 seconds which is a reduction of 86.5%. Overall, in all our experiments with single as well as multiple drives, we observed that the OPT algorithm achieved an average of 85% (random distribution) and 94% (hot-cold distribution) reduction in the average waiting times over the FCFS policy. It is interesting to note that despite the inefficiency of the naive FCFS policy, it is used in commercial systems [Moo96].

7 Conclusion

We have investigated the problem of efficient I/O scheduling for tertiary libraries, a problem of great importance to multimedia applications handling large volumes of data. We began by developing scheduling algorithms that minimize the number of media switches (*Minimum Switching* scheduling). Through experiments we established that for technologies such as optical disks which have switch times that dominate seek times, such schedules are likely to be optimal. For technologies that have more dominant seek times, such as tapes, it may be possible to get more efficient schedules by unloading media to service requests on other media before servicing all requests on the medium. However, such situations are rare and the gains are small. Moreover even for such media, the *Minimum Switching* scheduling policies show great improvement over more naive scheduling policies. We expect that the more complicated scheduling policies needed to achieve optimal scheduling will not achieve significant gains over the simpler *Minimum Switching* scheduling policies.

In the case of a single drive, we have developed a scheduling policy, OPT, that produces optimal schedules in $O(m \log m + N)$ time, where m is the number of media and N is the number of requests for all the media. OPT does not depend upon the order of processing of requests within a single medium. We can therefore choose the best available scheduling policy within a given medium independently. It does however, require knowledge of the total processing time required for each medium and the time that the library takes to switch media. In the absence of such information, an even simpler heuristic, *Number* has been proposed which also takes $O(m \log m + N)$ time and produces schedules that are very close to the optimal (within 1% for all our experiments).

In the presence of multiple drives, the problem of *Minimum Switching* scheduling becomes NP-Complete. Since no efficient solutions are expected to exist for NP-Complete problems, we proposed

two heuristics. The *OPT* algorithm, which gives optimal performance for the single drive problem, was found to be a very good heuristic for the multiple drive problem. It gave results within 1% of the lower bound for multiple drives. As in the single drive case, the even simpler *Number* heuristic was found to give schedules that are very close to those given by *OPT* for multiple drives.

We have therefore developed optimal and near-optimal solutions for the scheduling of I/O requests for robotic libraries. The algorithms are efficient and simple to implement and achieve significant improvements (85% - 94% for our workload) over the naive FCFS policy. They make no restrictions on the requests, the distribution of these requests, the seek or rewind times for the drives or the switch time for the library. Most importantly, the algorithms are applicable to any storage library that handles removable media, such as tapes and optical disks.

References

[ABZ96] M. Andrews, M. A. Bender, and L. Zhang. New algorithms for the disk scheduling problem. In *Foundations of Computer Science*, 1996.

[CDK+95] L. T. Chen, R. Drach, M. Keating, S. Louise, D. Rotem, and A. Shoshani. Efficient organization and access of multi-dimensional datasets on tertiary storage systems. In *Information Systems*, volume 20, pages 155–83. Elsevier Science, 1995.

[CHL93] M. J. Carey, L. M. Haas, and M. Livny. Tapes hold data, too: Challenges of tuples on tertiary store. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 413–7, Washington, DC, 1993.

[DK93] A. L. Drapeau and R. H. Katz. Striping in large tape libraries. In *Proc. of Supercomputing*, pages 378–387, Portland, Oregon, 1993. ACM.

[Exa96] Exabyte. Products. http://www.Exabyte.CO M:80/Products/, Oct. 1996.

[FM96] D. A. Ford and J. Myllymaki. A log-structured organization for tertiary storage. In *Proceedings of the Twelfth International Conference on Data Engineering*, pages 20–7, New Orleans, Louisiana, 1996.

[GJ79] M. R. Garey and D. S. Johnson. *Computers and Intractability, A Guide to the Theory of NP-Completeness.* W. H. Freeman and Company, New York, 1979.

[GM95] L. Golubchik and R. Muntz. Analysis of striping techniques in robotic storage libraries. In *Proceedings of the Fourteenth IEEE Symposium on Mass Storage Systems*, pages 225–38, Monterey, CA, 1995.

[HS96] B. K. Hillyer and A. Silberschatz. Random I/O scheduling in online tertiary storage. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, Canada, 1996.

[LKB77] J.K. Lenstra, A. K. G. Rinnooy Kan, and P. Brucker. Complexity of machine scheduling problems. *Annals of Discrete Mathematics*, 1:343–362, 1977.

[LLKS85] E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, and D. B. Shmoys. *The Traveling Salesman Problem.* Wiley, Chichester, 1985.

[LO96] J. Li and C. Orji. I/O scheduling in tape-based tertiary systems. *Journal of Mathematical Modelling and Scientific Computing*, 6, 1996.

[ML95] J. Myllymaki and M. Livny. Disk-tape joins: Synchronizing disk and tape access. In *Joint International Conference on Measurement and Modeling of ComputerSystems. SIGMETRICS '95/PERFORMANCE '95*, pages 279–90, Ottawa, Canada, 1995.

[ML96] J. Myllymaki and M. Livny. Efficient buffering for concurrent disk and tape I/O. In *Proceedings of Performance '96, Int. Conf. on Performance Theory, Measurement and Evaluation of Computer Communication Systems*, Lausanne, Switzerland, October 1996.

[Moo96] R. Moore. Private Communication, 1996.

[MP93] T. E. Morton and D. W. Pentico. *Heuristic Scheduling Systems with applications to Production Systems and Project Management.* John Wiley and Sons, Inc., 1993.

[PAES97] S. Prabhakar, D. Agrawal, A. El Abbadi, and A. Singh. Scheduling tertiary I/O in database applications. In *Proc. of the 8th International Workshop on Database and Expert Systems Applications*, pages 722–727, Toulouse, France, September 1997.

[Par95] G. R. Parker. *Deterministic Scheduling Theory.* Chapman & Hall, 2-6 Boundary Row, London SE1 8HN, UK, 1995.

[Pin95] M. Pinedo. *Scheduling Theory, Algorithms and Systems.* Prentice-Hall International Series in Industrial and Systems Engineering. Prentice-Hall, Englewood Cliffs, NJ, 1995.

[Qua96] Quantum. Products and technology. http://www.quantum.com/products, Oct. 1996.

[RW93] C. Ruemmler and J. Wilkes. UNIX disk access patterns. In *USENIX*, pages 405–420, Winter 1993.

[Sar95a] S. Sarawagi. Database systems for efficient access to tertiary memory. In *Proceedings of the Fourteenth IEEE Symposium on Mass Storage Systems*, pages 120–6, Monterey, California, 1995.

[Sar95b] S. Sarawagi. Query processing in tertiary memory databases. In *Proc. of the 21st Int. Conf. on Very Large Data Bases*, pages 585–596, San Francisco, California, 1995. Morgan Kaufmann.

[Sch86] H. D. Schwetman. CSIM: A C-based, process-oriented simulation language. In *Proceedings of the 1986 Winter Simulation Conference*, pages 387–396, December 1986.

[SON96] SONY. Recording media products. http://sony.sosin.com.sg/record-media/, Oct. 1996.

[SS93] S. Sarawagi and M. Stonebraker. Single query opimization for tertiary memory. Technical Report s2k-94-45, Computer Science Div. U.C. Berkeley, December 1993.

[SS94] S. Sarawagi and M. Stonebraker. Efficient organization of large multidimensional arrays. In *IEEE Int. Conf. on Data Engineering*, pages 328–336, Houston, TX, USA, Feb. 1994.

[Teo72] T. J. Teorey. Properties of disk scheduling policies in multiprogrammed computer systems. In *Proceedings of the AFIPS Fall Joint Computer Conference*, pages 1–11, 1972.

Appendix A: Derivation of Total Waiting Time for Single Drive

In the following discussion the ordering of the media refers to the sequence in which they are loaded onto the drive. The following notation is used:

- w_{ij} denotes the amount of time that the jth request (in the order of execution on the medium, as determined by the scheduling algorithm) on medium i waits before transfer begins.
- s_{ij} denotes the time required to seek from the end of the $(j-1)$th request to the first block of the jth request on medium i. For $j = 1$, it is the time required to seek from the initial position of the head after loading to the first request.
- T is the time required to switch a medium. It includes the time to eject the currently loaded medium, time for the robot to pick the ejected medium, place it on shelf move to the next medium, put the new medium into the drive and time for the drive to load the new medium (we assume that this time is constant, independent of the location of the drive or medium, as given in most product specifications).
- n_i denotes the number of requests for medium i.
- x_{ij} is the time required to transfer data for the jth request on medium i.
- r_i denotes the time required to rewind medium i after all requests for the medium have been serviced. For optical disks this time may be absent or negligible. For tapes it is the time required to rewind to the beginning of the tape and the time to write the control information.
- W_i denotes the total waiting time for all requests on medium i.
- m is the number of media with requests.

The waiting times for requests of the first medium are as follows:

$$\begin{aligned} w_{11} &= T + s_{11} \\ w_{12} &= T + s_{11} + x_{11} + s_{12} \\ &\vdots \\ w_{1n_1} &= T + s_{11} + x_{11} + s_{12} + x_{12} + \ldots + x_{1n_1-1} + s_{1n_1} \end{aligned}$$

The sum of the waiting times for requests for the first medium, W_1, is therefore:

$$W_1 = n_1 \cdot T + \sum_{j=1}^{n_1} (n_i - j + 1) \cdot s_{1j} + \sum_{j=1}^{n_1-1} (n_i - j) \cdot x_{1j}$$

The waiting times for the second medium are:

$$\begin{aligned} w_{21} &= Q_1 + T + s_{21} \\ w_{22} &= Q_1 + T + s_{21} + x_{21} + s_{22} \\ &\vdots \\ w_{2n_2} &= Q_1 + T + s_{21} + x_{21} + \ldots + x_{2n_2-1} + s_{2n_2} \end{aligned}$$

where $Q_1 = T + \sum_{j=1}^{n_1} s_{1j} + \sum_{j=1}^{n_1} x_{1j} + r_1$, is the time spent before the second medium is scheduled to be loaded. It includes the switch time for the first medium, the seek and transfer times for each request of the first medium and the time for rewinding the medium. The total time for the second medium, W_2 is:

$$W_2 = n_2 \cdot Q_1 + T) + \sum_{j=1}^{n_2} (n_2 - j + 1) \cdot s_{2j} + \sum_{j=1}^{n_2-1} (n_2 - j) \cdot x_{2j}$$

$$W_2 = 2T \cdot n_2 + n_2 (\sum_{j=1}^{n_1} s_{1j} + \sum_{j=1}^{n_1} x_{1j} + r_1) + \sum_{j=1}^{n_2} (n_2 - j + 1) s_{2j} + \sum_{j=1}^{n_2-1} (n_2 - j) x_{2j}$$

Similarly, for the ith medium,

$$W_i = n_i \cdot Q_{i-1} + T) + \sum_{j=1}^{n_i} (n_i - j + 1) \cdot s_{ij} + \sum_{j=1}^{n_i-1} (n_i - j) \cdot x_{ij}$$

where $Q_{i-1} = (i-1)T + \sum_{k=1}^{i-1} \sum_{j=1}^{n_k} s_{kj} + \sum_{k=1}^{i-1} \sum_{j=1}^{n_k} x_{kj} + \sum_{k=1}^{i-1} r_k$

$$W_i = i \cdot T \cdot n_i + n_i \left(\sum_{k=1}^{i-1} \left(\sum_{j=1}^{n_k} (s_{kj} + x_{kj}) + r_k \right) \right) + \sum_{j=1}^{n_i} (n_i - j + 1) s_{ij} + \sum_{j=1}^{n_i-1} (n_i - j) x_{ij}$$

The total waiting time for all requests, W, is therefore:

$$W = \sum_{i=1}^{m} W_i = \sum_{i=1}^{m} \left(\sum_{j=1}^{n_i} (n_i - j + 1) s_{ij} + \sum_{j=1}^{n_i-1} (n_i - j) x_{ij} \right) + \sum_{i=1}^{m} i \cdot T \cdot n_i + \sum_{i-1}^{m} \left(n_i \left(\sum_{k=1}^{i-1} \left(\sum_{j=1}^{n_k} (s_{kj} + x_{kj}) + r_k \right) \right) \right)$$

Let $P_k = \sum_{j=1}^{n_k} (s_{kj} + x_{kj}) + r_k$, the processing time for tape i, and
$C_i = \sum_{j=1}^{n_i} (n_i - j + 1) s_{ij} + \sum_{j=1}^{n_i-1} (n_i - j) x_{ij}$, the total waiting time incurred by the requests on medium i after medium i has been loaded onto the drive.
Therefore,

$$W = \sum_{i=1}^{m} C_i + \sum_{i=1}^{m} T i n_i + \sum_{i=1}^{m} \left(n_i \sum_{k=1}^{i-1} P_k \right)$$

which can be rewritten as:

$$W = \sum_{i=1}^{m} C_i + \sum_{i=1}^{m} T \cdot i \cdot n_i + \sum_{i=1}^{m} \left(n_i \sum_{k=1}^{i-1} P_k \right)$$

Appendix B: Proof of Lemma 3.1

First, we establish the following Lemma:

Lemma B.1 *In any schedule, if two adjacent media, i and $i+1$ are exchanged, the change in the total waiting time for the schedule is given by:*

$$\Delta W = n_i T + n_i P_{i+1} - n_{i+1} T - n_{i+1} P_i$$

Proof: By Property 1 only the requests on media i and $i+1$ will change as a result of the exchange. Each request on medium i sees an increase in the waiting time due to the switch. The increase is an extra medium switch and the processing time of medium $i+1$. Therefore the increase in waiting time is: $n_i T + n_i P_{i+1}$. Similarly, each request on medium $i+1$ sees a decrease in the waiting time. The decrease is equal to one medium switch and the processing time for medium i (since medium i no longer precedes it in the new order). The amount of the decrease for medium $i+1$ is: $n_{i+1} T + n_{i+1} P_i$. Hence the total change in the total waiting time is:

$$n_i T + n_i P_{i+1} - n_{i+1} T - n_{i+1} P_i.$$

□

Proof of Lemma 3.1: Let us assume that for an optimal solution, there exists k, $1 \leq k < m$, such that the above condition is not true, i.e.,

$$n_k (T + P_{k+1}) < n_{k+1} (T + P_k)$$

Consider a schedule which differs from $O_1, O_2, \cdots, O_m$ only in that the media O_k and O_{k+1} are exchanged. Therefore by Lemma B.1 the total waiting time for such a schedule would differ from that of the optimal schedule by

$$\Delta W = n_k (T + P_{k+1}) - n_{k+1} (T + P_k)$$

which is negative. Hence we see that the new schedule has a lower total waiting time than the optimal schedule $O_1, O_2, \cdots, O_m$. Therefore this schedule could not be optimal, leading to a contradiction. Hence our assumption that there exists k which satisfies $n_k (T + P_{k+1}) < n_{k+1} (T + P_k)$ must be false. Therefore $\forall k \;\; 1 \leq k < m, \;\; n_k (T + P_{k+1}) \geq n_{k+1} (T + P_k)$.
Rearranging, we get

$$\forall k,\, 1 \leq k < m,\; \frac{n_k}{T + P_k} \geq \frac{n_{k+1}}{T + P_{k+1}}$$

□

Appendix C: Derivation of Total Waiting Time for Multiple Drives

For each drive, Equation 1 developed in Section 3 gives the total waiting times. The total waiting time for all media processed on drive l is:

$$W^l = \sum_{i=1}^{m^l} \left(C_i^l + T \cdot i \cdot n_i^l + n_i^l \sum_{j=1}^{i-1} P_j^l \right)$$

where

- m^l is the number of media that are processed on drive l
- C_i^l is the constant component of the wait time for requests on the ith medium on drive l
- n_i^l is the number of requests for the ith medium on drive l
- $T \cdot i \cdot n_i^l$ is the wait time for requests of the ith medium on drive l due to the switching of preceding media in drive l
- P_j^l is the processing time for the ith medium on drive l
- $n_i^l \sum_{j=1}^{i-1} P_j^l$ is the wait time for requests on the ith medium on drive l due to the processing of preceding media on drive l.

Assuming d drives, the total waiting time for all media is given by,

$$W = \sum_{k=1}^{d} \sum_{i=1}^{m^k} \left(C_i^k + T \cdot i \cdot n_i^k + n_i^k \sum_{j=1}^{i-1} P_j^k \right) \tag{8}$$

This can be rewritten as:

$$\begin{aligned} W &= \sum_{k=1}^{d} \sum_{i=1}^{m^k} C_i^k + \sum_{k=1}^{d} \sum_{i=1}^{m^k} \left(n_i^k \cdot T \cdot i + n_i^k \sum_{j=1}^{i-1} P_j^k \right) \\ &= \sum_{k=1}^{d} \sum_{i=1}^{m^k} C_i^k + \sum_{k=1}^{d} \sum_{i=1}^{m^k} \left(n_i^k \sum_{j=1}^{i-1} (T + P_j^k) + T n_i^k \right) \\ &= \sum_{k=1}^{d} \left(\sum_{i=1}^{m^k} C_i^k + \sum_{i=1}^{m^k} (T \cdot n_i^k) \right) + \sum_{k=1}^{d} \sum_{i=1}^{m^k} \left(n_i^k \sum_{j=1}^{i-1} (P_i^k + T) \right) \end{aligned} \tag{9}$$

The first term in Equation 9 is the sum of the constant times for all media ($\sum_{k=1}^{d} \sum_{i=1}^{m^k} C_i^k$) and the product of the switch time, T and the sum of the number of requests for all media ($T \sum_{k=1}^{d} \sum_{i=1}^{m^k} n_i^k$). Both these factors are independent of the order of processing of media. Therefore, the total waiting time can be written as:

$$W = K + \sum_{k=1}^{d} \sum_{i=1}^{m^k} \left(n_i^k \sum_{j=1}^{i-1} (P_i^k + T) \right) \tag{10}$$

where $K = \sum_{k=1}^{d} \sum_{i=1}^{m^k} \left(C_i^k + T \cdot n_i^k \right)$ is a constant independent of the assignment of media to drives or the order of processing of media on drives.

Appendix D: Proof of NP-Completeness of Multidrive Minimum Switching Scheduling

Proof: We will prove Theorem 4.1 by showing that a known NP-Complete problem can be reduced to the above problem ($MMSS$). We will reduce the problem of *Scheduling to Minimize Weighted Completion Time ($MWCT$)* [LKB77], which is defined as follows, to $MMSS$.

Definition D.1 Scheduling to Minimize Weighted Completion Time (MWCT) GIVEN: *Set S of tasks, number $m \in Z^+$ of processors, for each task $t \in S$ a length $l(t) \in Z^+$ and a weight $w(t) \in Z^+$, and a positive integer J'.*
QUESTION: *Is there an m-processor schedule σ for S such that the sum, over all $t \in S$, of $(\sigma(t) + l(t)) \cdot w(t)$ is no more than J'?*

The $MWCT$ problem has been shown to be NP-Complete in [LKB77]. Given any instance of the $MWCT$ problem, we create an instance of the $MMSS$ problem as follows. For each task $t \in S$ of $MWCT$, we create a medium in the $MMSS$ problem such that the number of requests for the medium is equal to the weight of t and the processing time for the medium is equal to the length of the task t. The number of drives for the $MMSS$ problem is set equal to the number of processors in the $MWCT$ problem. The value of the constant J for the $MMSS$ problem is set to be equal to the constant for the $MWCT$ problem, J', plus K', where K' is defined above as the constant factor in the total waiting time in Equation 4. Now we see that any schedule that satisfies the $MMSS$ problem will generate a schedule that results in a value of W which is smaller than or equal to J. If from this schedule we derive a schedule for the $MWCT$ problem by simply substituting the corresponding tasks and processors for the media and drives, we will get a schedule which has a total weighted completion time of no more than $J - K'$, which is equal to J'. Similarly, any schedule for the $MWCT$ problem which results in a weighted completion time no more than J' can be easily converted into a schedule for $MMSS$ which has a total waiting time of no more than $J' + K'$, which is equal to J. Hence, a solution for $MWCT$ exists if and only if a solution for $MMSS$ exists. Clearly, $MMSS$ is in NP because we can guess a schedule and determine it's total completion time and compare it with J in polynomial time. Since $MMSS$ is in NP and an NP-Complete problem ($MWCT$) can be reduced to $MMSS$, we conclude $MMSS$ is an NP-Complete problem. □