Theory Comput. Systems **31**, 491–506 (1998)

Theory of Computing Systems

Competitive Analysis of On-Line Disk Scheduling*

Tzuoo-Hawn Yeh, Cheng-Ming Kuo, Chin-Laung Lei, and Hsu-Chun Yen

Department of Electrical Engineering, National Taiwan University,
Taipei, Taiwan, Republic of China

Abstract. In this paper we study three popular on-line disk scheduling algorithms, FCFS, SSTF, and LOOK, using *competitive analysis*. Our results show that, in a competitive sense, the performance of LOOK is better than those of SSTF and FCFS. As a by-product, our analysis also reveals quantitatively the role played by the size of the *window*, which in our model is a waiting buffer that holds a fixed number of requests waiting to be serviced next. The window, in some sense, offers the *lookahead* ability which is mentioned in several on-line problems.

1. Introduction

Disk scheduling is a problem that is of practical importance and theoretical interest in the study of computer systems, in particular, in the areas of databases and operating systems. The goal of disk scheduling is to devise a policy for servicing disk requests in an on-line fashion so as to minimize the total disk access time. Traditionally, measuring the performance of a disk scheduling algorithm often resorts to either *probabilistic analysis* [4], [7], [14], [15] or *simulation* [6], [7], [14], [15]. In spite of their popularity, the above approaches have their limitations. For example, coming up with an accurate probability distribution for the disk access pattern is difficult, yet such a distribution is critical in measuring the average-case performance of disk scheduling. In addition, neither of them aims for reporting the worst-case performance of an on-line algorithm. In an attempt to overcome the above shortcomings, *amortized analysis*, which is a technique known to be useful for measuring the worst-case complexity of performing a sequence of operations, has recently been applied to analyzing a number of disk scheduling algorithms [3]. It, however, provides no information regarding the relative performance of an algo-

* This work was supported in part by the National Science Council of the Republic of China under Grant NSC-84-2213-E-002-002.

rithm against another one such as, for example, an optimal off-line algorithm which has complete knowledge of future disk requests.

As an alternative to the aforementioned analytical techniques, in this paper we study various disk scheduling algorithms through the use of the so-called *competitive analysis* technique. Competitive analysis has its origin in a paper by Sleator and Tarjan [13]. In this approach, the efficiency of an on-line algorithm is compared with that of an optimal off-line algorithm. An on-line algorithm is *α-competitive* if, for any request sequence, its cost is within α times that of the optimal off-line algorithm, up to a constant that is independent of the request sequence. For more about competitive analysis on other problems, see, e.g., [1], [2], [5], [9], and [11].

Given a sequence of requests on a disk equipped with a single read-write head and a *waiting buffer* of a fixed size, the goal of disk scheduling is to design a strategy for picking requests, in an on-line fashion, from the waiting buffer to service so that the total cost of service is minimized. The waiting buffer, referred to as the *window* in this paper, is capable of holding a fixed amount of requests from which the next request to service is chosen. Once a request enters the window, it is entitled to be serviced next even though the window might contain requests that arrived earlier. Conversely, a request can be serviced only if it resides in the window. In our model, disk scheduling algorithms are differentiated mainly by the order in which the next request to service is chosen. Notice that there are no deadlines for servicing requests. In particular, it is possible that the first request will be serviced last, no matter how long the sequence is. To a certain degree, the window offers the lookahead capability, mentioned in several on-line problems [8], [10]. As our results show, the window is beneficial to disk scheduling algorithms.

The time to satisfy a disk request is composed of the *seek* time, the *latency* time, and the *transfer* time [12]. The seek time is usually approximated as $s + d \cdot t$, where s is the *seek startup* time, d is the seek distance, and t is the seek time per cylinder [6], [14], [15]. In this paper we use a model in which the cost of servicing a request is measured as $d + T$, where d is the seek distance, measured in terms of the number of cylinders, and T is a constant that captures the seek startup time, the latency time, and the transfer time.

We consider the following three on-line disk scheduling algorithms: FCFS, SSTF, and LOOK. FCFS (first come first serviced) services requests according to their original order. SSTF (shortest seek time first) chooses the request in the window that is closest to its disk head to service. LOOK partitions the request sequence into blocks of W requests each. Requests in a block are serviced completely, with minimum cost, before the ones in later blocks. That is, let l and r be the leftmost and rightmost requests, respectively, in the block and assume that l is closer to the disk head than r is, LOOK moves the disk head to l then to r to service the requests in the block. The case where r is closer is symmetrical. In this paper we prove that the lower bounds of the competitive ratios of FCFS, SSTF, and LOOK are

$$\frac{T + D}{T + D/(2W - 1)}, \qquad \frac{T + \lfloor (D-1)/2 \rfloor}{T + \lfloor (D-1)/2 \rfloor/(2W - 1)}, \quad \text{and}$$

$$\frac{WT + D}{WT + D/(2W - 1)},$$

respectively, where D is the number of cylinders in the disk less one, W is the size of the

window, and T is the constant mentioned in our cost model. We also give a competitive ratio of LOOK that matches its lower bound. As our results show, the performance of LOOK, in a competitive sense, is better than those of FCFS and SSTF.

The paper is organized as follows. Section 2 gives the disk model, the cost measure, the technique of competitive analysis, and the definitions of FCFS, SSTF, and LOOK. Sections 3 concerns itself with deriving the lower bounds of the competitive ratios of FCFS, SSTF, and LOOK. In Sections 4 and 5 we derive the competitive ratio of LOOK using the potential technique. Sections 6 gives the discussions and concluding remarks.

2. Preliminaries

A *disk* consists of a disk head, a window, and $D + 1$ cylinders numbered from 0 to D. The *disk head* is a server that moves on the disk and services requests on cylinders. The disk is usually regarded as a line segment where cylinders 0 and D are the leftmost and rightmost cylinders, respectively, on the disk. The distance between cylinders r_1 and r_2 is $|r_1 - r_2|$. A *request* on cylinder r is denoted as $\langle r \rangle$. A *request sequence* is a sequence of requests, waiting to be serviced by the disk head. A request sequence $\sigma = \langle r_1, r_2, \ldots, r_n \rangle$ specifies a sequence of n requests on cylinders r_1, r_2, ..., and r_n. The *window* is a buffer of size W which is capable of holding W requests. Requests in σ enter the window one-by-one, and a request must be in the window before it is serviced. Without loss of generality, we assume that the window is always full unless there is no request in the remaining request sequence. That is, at each step of service, one request is serviced and one is loaded into the window, if there are requests remaining in the request sequence. An *algorithm* controls the movement of the disk head and decides which request in the window is to be serviced next. The algorithm moves the disk head to cylinder r to service $\langle r \rangle$. (Usually, we use the phrase "the algorithm moves to r to service $\langle r \rangle$" as a shorthand for the previous one.) Consider an algorithm A. The *schedule* of A on σ, denoted as $\pi_{\mathsf{A}}(\sigma) = \langle\langle x_1, x_2, \ldots, x_n \rangle\rangle$, is the servicing sequence of A on σ. Notice that $\pi_{\mathsf{A}}(\sigma)$ is a permutation of σ and x_i, $1 \leq i \leq n$, is the ith serviced request. Let $h = x_0$ be the initial disk head position of A. Without loss of generality, we assume that A moves straight from x_{i-1} to x_i to service $\langle x_i \rangle$, $i \leq 1 \leq n$. The *path* of A on σ is a sequence of cylinders that records the traveling history of A. The *cost* of A on σ, denoted as $\text{cost}_{\mathsf{A}}(\sigma)$, is defined to be $\sum_{i=1}^{n} |x_i - x_{i-1}| + nT$, where T is a constant that captures the seek startup time, the latency time, and the transfer time of servicing a request.

An algorithm is *on-line* if the decision of which request to service next depends only on its past history and the current window content. That is, the decision is independent of the remaining requests that have not been brought into the window. Otherwise, the algorithm is *off-line*. That is, an off-line algorithm is aware of the whole request sequence when it makes decisions. The *on-line disk scheduling problem* is that of designing an on-line algorithm to service request sequences with minimum cost.

An on-line algorithm A is α*-competitive* if there exists a constant β such that, for any request sequence σ and any algorithm ADV,

$$\text{cost}_{\mathsf{A}}(\sigma) \leq \alpha \times \text{cost}_{\mathsf{ADV}}(\sigma) + \beta.$$

The ratio α is called the *competitive ratio* of A. Notice that, since one of the algorithms

represented by ADV is an optimal algorithm on σ, the definition of competitiveness above is comparing A to an optimal off-line algorithm. The definition given here can be viewed as a game between the on-line algorithm A and the adversary ADV. ADV is responsible for generating a worst-case sequence against A, and servicing the sequence with minimum cost. Notice that, when $W = 1$, the competitive ratio of any algorithm is 1, since there is exactly one request in the window to chose from, and the behavior of any algorithm, no matter whether it is A or ADV, is exactly the same. As a result, we assume that $W \geq 2$ in the remainder of this paper.

In this paper we consider the following three on-line algorithms:

FCFS: Given a request sequence σ, FCFS services σ using the schedule $\langle\langle\sigma\rangle\rangle$.

SSTF: SSTF chooses the request in the window that is nearest to the disk head to service. Break tie arbitrarily.

LOOK: The request sequence is partitioned into blocks, each of which, except perhaps the last one, contains W requests. The requests in a block are serviced before the requests in the next block. Let V be the block of requests under consideration. Assume that $\langle l\rangle$ and $\langle r\rangle$ are the smallest and largest requests, respectively, in V, that is, $l \leq q \leq r$ for any $\langle q\rangle \in V$. Let h be the disk head position before the current block is being serviced. If $|l - h| < |r - h|$, then the disk head moves to l and sweeps to r, services all requests in V. Symmetrically, if $|r - h| > |l - h|$, then the disk head moves to r and sweeps to l, services all requests in V. Break tie arbitrarily.

3. Lower Bounds

In this section the lower bounds of the competitive ratios for FCFS, SSTF, and LOOK are derived. Initially, the disk head is assumed to be on cylinder 0. For ease of expression, let σ^k represent σ repeated k times, where σ is a request or a sequence of requests. To show that α_{LB} is a lower bound of the competitive ratio of an on-line algorithm A, we proceed as follows. For any constant β, construct a request sequence σ, according to the behavior of A, such that

$$\frac{\mathrm{cost}_{\mathsf{A}}(\sigma) - \beta}{\mathrm{cost}_{\mathsf{ADV}}(\sigma)} \geq \alpha_{\mathrm{LB}}.$$

In the following derivation, let n be a large integer.

Theorem 1. *The competitive ratio of* FCFS *is no less than*

$$\frac{T + D}{T + D/(2W - 1)}.$$

Proof. Consider the request sequence:

$$\sigma = \langle (D0)^{(2W-1)n}\rangle.$$

FCFS services σ using the schedule $\langle\langle\sigma\rangle\rangle$, so $\text{cost}_{\text{FCFS}}(\sigma) = 2n(2W-1)(T+D)$. We can service σ more efficiently as follows. First we stay on cylinder 0, service as many requests on cylinder 0 as possible, delay the service of the requests on cylinder D by accumulating them in the window, until the window is full of requests on cylinder D. Then we move to cylinder D and service the W requests on cylinder D in the window. We can repeat symmetrically the process described above on cylinder D, and finally go back to cylinder 0. The process can be repeated until all the requests in σ are processed. That is, we can service σ using the schedule

$$\langle\langle(0^{W-1}D^{2W-1}0^{W})^{n}\rangle\rangle,$$

so $\text{cost}_{\text{ADV}}(\sigma) \leq 2n(2W-1)T + 2nD$. Thus,

$$\frac{\text{cost}_{\text{FCFS}}(\sigma) - \beta}{\text{cost}_{\text{ADV}}(\sigma)} \geq \frac{2n(2W-1)(T+D) - \beta}{2n(2W-1)T + 2nD},$$

which approaches

$$\frac{T+D}{T + D/(2W-1)}$$

when n is large. □

Theorem 2. *The competitive ratio of* SSTF *is no less than*

$$\frac{T + \lfloor (D-1)/2 \rfloor}{T + \lfloor (D-1)/2 \rfloor/(2W-1)}.$$

Proof. Consider the following request sequence:

$$\sigma = \left\langle D^{W-1}\left(\left\lfloor \frac{D-1}{2} \right\rfloor 0\right)^{n(2W-1)+W}\right\rangle.$$

SSTF services σ using the schedule

$$\left\langle\left\langle\left(\left\lfloor \frac{D-1}{2} \right\rfloor 0\right)^{n(2W-1)+W} D^{W-1}\right\rangle\right\rangle,$$

so $\text{cost}_{\text{SSTF}}(\sigma) = (2n(2W-1) + 3W - 1)T + 2(n(2W-1)+W)\lfloor (D-1)/2 \rfloor + D$. We can service σ more efficiently as follows. First we move to cylinder D and service D^{W-1}. Then we service the remaining requests in σ using technique similar to the process carried out in the previous theorem. That is, we can service σ using the schedule

$$\left\langle\left\langle D^{W-1}\left\lfloor \frac{D-1}{2} \right\rfloor^{W} 0^{W}\left(0^{W-1}\left\lfloor \frac{D-1}{2} \right\rfloor^{2W-1} 0^{W}\right)^{n}\right\rangle\right\rangle,$$

so $\text{cost}_{\text{ADV}}(\sigma) \leq (2n(2W-1)+3W-1)T+2D+2n\lfloor(D-1)/2\rfloor$. When n is large, we have

$$\frac{\text{cost}_{\text{SSTF}}(\sigma)-\beta}{\text{cost}_{\text{ADV}}(\sigma)} \geq \frac{T+\lfloor(D-1)/2\rfloor}{T+\lfloor(D-1)/2\rfloor/(2W-1)}.$$

□

Theorem 3. *The competitive ratio of* LOOK *is no less than*

$$\frac{WT+D}{WT+D/(2W-1)}.$$

Proof. Consider the request sequence:

$$\sigma = \langle((0^{W-1}D0^W)^{W-1}D^W \cdot (D^{W-1}0D^W)^{W-1}0^W)^n\rangle.$$

LOOK services σ using the schedule $\langle\langle\sigma\rangle\rangle$, so $\text{cost}_{\text{LOOK}}(\sigma) = 2nW(2W-1)T+2n(2W-1)D$. We can service σ more efficiently as follows. First we stay on cylinder 0 and service all the requests on cylinder 0 in the subsequence $\langle(0^{W-1}D0^W)^{W-1})\rangle$. The $W-1$ requests on cylinder D in the subsequence is delayed in the window. Then we move to cylinder D, service the $W-1$ requests on cylinder D left in the window, and service all the requests on cylinder D in the subsequence $\langle(D^{W-1}0D^W)^{W-1})\rangle$. The $W-1$ requests on cylinder 0 in the subsequence is delayed in the window. Then we go back to cylinder 0 and service the $W-1$ requests on cylinder 0 delayed in the window. The process can be repeated for n times. That is, we can service σ using the schedule

$$\langle\langle(0^{(W-1)(2W-1)}D^{2W-1} \cdot D^{(W-1)(2W-1)}0^{2W-1})^n\rangle\rangle,$$

so $\text{cost}_{\text{ADV}}(\sigma) \leq 2nW(2W-1)T+2nD$. When n is large, we have

$$\frac{\text{cost}_{\text{LOOK}}(\sigma)-\beta}{\text{cost}_{\text{ADV}}(\sigma)} \geq \frac{WT+D}{WT+D/(2W-1)}.$$

□

4. The Behavior of LOOK and ADV

In this and the next sections we prove the competitiveness of LOOK that matches its lower bound. In this section we give the concept of a "phase," which corresponds to the period of time that LOOK services a block of W requests. To analyze LOOK, we run LOOK and ADV phase by phase in parallel. We show that the behavior of LOOK and ADV in any phase can be transformed into a simpler form and give some properties about it. In the next section we analyze the simpler form using the potential function technique.

Let $\sigma = \langle r_1, r_2, \ldots, r_n\rangle$ be a request sequence of length n and $p = \lceil n/W\rceil$. The ith phase of requests, denoted as δ_i, $1 \leq i \leq p-1$, is the sequence $\delta_i = \langle r_{(i-1)W+1}, r_{(i-1)W+2}, \ldots, r_{iW}\rangle$. The last phase of requests, i.e., the pth phase of requests, is $\delta_p = \langle r_{(p-1)W+1}, \ldots, r_n\rangle$. Intuitively, σ is partitioned into p phases of requests, each of which, except perhaps the last one, contains exactly W requests. The ith phase of

LOOK, $1 \leq i \leq p$, is the time period where LOOK services the ith phase of requests. Since ADV is an arbitrary algorithm and is allowed to delay the service of some requests indefinitely until the end of its schedule, the notion of a phase of ADV is different. Let $\pi_{\mathsf{ADV}}(\sigma) = \langle\langle x_1, x_2, \ldots, x_n \rangle\rangle$. The first phase of ADV is defined to be the time period where ADV services $\langle x_1 \rangle$. From the second phase on, ADV services W requests in each phase. In the last phase, ADV services less than or equal to W requests. Notice that the number of phases for ADV may be p or $p+1$. More precisely, if $n = (p-1)W + 1$, then ADV has p phases, and in the pth phase ADV services W requests. Otherwise, that is, $(p-1)W + 2 \leq n \leq pW$, ADV has $p+1$ phases, and in the $(p+1)$th phase ADV services $n - (p-1)W - 1$ requests. Notice that LOOK services the requests in δ_i in the ith phase and ADV services the requests in δ_i in the ith or later phases.

Define the *delayed set* of ADV at the end of phase i, $1 \leq i \leq p-1$, to be the $W-1$ requests in $\langle \delta_1, \delta_2, \ldots, \delta_i \rangle$ that have not yet been serviced by ADV. Intuitively, the delayed set equals the window content minus the request that is brought into the window last. Define the *configuration* of the disk at the end of phase i, $1 \leq i \leq p-1$, to be the three-tuple $(A, L, \mathcal{B})$, where A and L are the disk head positions of ADV and LOOK, and $\mathcal{B}$ is the delayed set of ADV.

Consider the behavior of LOOK and ADV in the ith phase, $2 \leq i \leq p-1$, and call this phase the "current phase." The first, the pth, and the $(p+1)$th, if it exists, phases are excluded from the following analysis because LOOK and/or ADV may service less than W requests in these phases. It will be clear in the next section where the competitive analysis is carried out that the exclusion of these phases does not change the desired result. Let $\delta = \langle s_1, s_2, \ldots, s_W \rangle$ be the current phase of requests, let $(A, L, \mathcal{B})$ be the configuration of the disk at the beginning of the phase, that is, the initial configuration, and let $(A', L', \mathcal{B}')$ be the configuration of the disk at the end of the phase, that is, the final configuration. Notice that in the current phase LOOK services all the W requests in δ, ADV services W of the $2W-1$ requests in $\mathcal{B} \cup \delta$, and $\mathcal{B}'$ equals the set of the $W-1$ requests in $\mathcal{B} \cup \delta$ that are not serviced by ADV. Table 1 lists all possible cases where $L \leq L'$ that describe the behavior of LOOK and ADV during the current phase. (The cases where $L > L'$ are symmetrical.) A general case can be viewed as a transition

Table 1. The general cases.

Case	Description
GC1 $A < L \leq L'$	
GC2 $L \leq A \leq L'$	$L \leq s_i \leq L'$ for all i and there exists an integer j such that $s_j = L'$. LOOK moves to L' and services δ.
GC3 $L \leq L' < A$	
GC4 $A \leq l$	
GC5 $l < A < L'$	$l \leq s_i \leq L'$ for all i and $s_{j_1} = l$ and $s_{j_2} = L'$ for two integers j_1 and j_2. LOOK moves to $l < L$ then to $L' > L$ and services δ. Notice that $(L' - L) \geq (L - l)$.
GC6 $L' \leq A$	

Table 2. The primitive cases and the virtual move of the adversary.

Case	Description
PC1	$A = L$ and $\delta = \langle A^{W-1}L' \rangle$ for some $L' \geq L$. LOOK moves to L' and services δ. ADV services $\langle A^{W-1} \rangle$, moves arbitrarily on the disk, settles on cylinder A', and services $\langle A' \rangle$. Assume that $\langle A' \rangle$ is the first request in $\mathcal{B} \cup \{\langle L' \rangle\}$ that ADV encounters during its move.
PC2	$A \geq L$ and $\delta = \langle A^{W} \rangle$. LOOK moves to A and services δ. ADV stays on A and services δ.
PC3	$\delta = \langle A^{W-2}lL' \rangle$ for some l and L' such that $l < A < L'$, $l < L < L'$, and $(L' - L) \geq (L - l)$. LOOK moves to l then to L' and services δ. Assume that there exist two requests $\langle q_1 \rangle, \langle q_2 \rangle \in \mathcal{B}$ such that $q_1 = q_2 = A$. ADV stays on A and services $\langle A^{W} \rangle$.
PC4	$A > L$ and $\delta = \langle A^{W-1}l \rangle$ for some $l < L$ such that $(A - L) \geq (L - l)$. LOOK moves to l then to A and services δ. ADV services $\langle A^{W-1} \rangle$, moves arbitrarily on the disk, settles on cylinder A', and services $\langle A' \rangle$. Assume that $\langle A' \rangle$ is the first request in $\mathcal{B} \cup \{\langle l \rangle\}$ that ADV encounters during its move.
VM	ADV moves arbitrarily on the disk, and replaces each request in the delayed set that is visited during the move with an arbitrary request. LOOK stays on its original position.

from the initial configuration to the final configuration. Since it is difficult to analyze the general cases directly, we simulate the transition of configurations in a general case with a sequence of simple cases listed in Table 2. The first four simple cases in the table are called *primitive cases*. They are cases extracted from general cases that are easier to handle. The last case in the table is the *virtual move of the adversary*. It is an artificial case and is not an ordinary operation found in a phase like the general cases and the primitive cases. In a virtual move, LOOK does not move in the transition of configurations. In the following, we try to *simulate* the transition from one configuration to another in a general case with a sequence of primitive cases and/or virtual moves.

Lemma 1. *Given any general case G, there is a sequence $P = P_1 P_2 \cdots P_v$ of $v \geq 1$ steps, where P_i, $1 \leq i \leq v$, is a primitive case or a virtual move such that* (1) *the initial configuration of G equals the initial configuration of P_1*; (2) *the final configuration of P_i equals the initial configuration of P_{i+1}, for $1 \leq i \leq v - 1$*; (3) *the final configuration of G equals the final configuration of P_v*; (4) *the path of* LOOK *in G equals that in P*; *and* (5) *the path of the adversary in G equals that in P.*

The lemma implies that, when $T = 0$, the cost of LOOK (resp. the adversary) in G equals the cost of LOOK (resp. the adversary) in P.

Proof. Let $(A_{(i)}, L_{(i)}, \mathcal{B}_{(i)})$ be the configuration of the disk after Step P_i. Imagine that an adversary ADVB is responsible for the construction of P. First we assume that G fits Case GC1 and analyze the behavior of LOOK and ADV in Case GC1. LOOK moves from L to L' and services δ. ADV moves arbitrarily on the disk. All we know is that,

since ADV services W requests in $\mathcal{B} \cup \delta$, ADV services at least one request in δ. As a result, ADV must visit cylinder L. The construction of P for Case GC1 consists of three steps, P_1, P_2, and P_3, each of which is a primitive case or a virtual move, as in the following.

*Case GC*1

Step P_1. ADVB simulates the path of ADV until it reaches cylinder L, and replaces each request in its delayed set visited during the move with a request on cylinder L. This step fits Case VM.

Step P_2. ADVB gives the request sequence $\langle L^{W-1}L' \rangle$. LOOK moves to L' and services $\langle L^{W-1}L' \rangle$. ADVB services $\langle L^{W-1} \rangle$, then simulates the remaining path of ADV until it reaches a cylinder q such that $\langle q \rangle \in \mathcal{B}_{(1)} \cup \{\langle L' \rangle\}$, and then services $\langle q \rangle$. This step fits Case PC1. Notice that if ADV services all the requests in δ in G, then L' is in its path; if ADVB replaced at least one request in Step P_1, then $\langle L \rangle \in \mathcal{B}_{(1)}$; otherwise, ADV services at least one request $\langle x \rangle \in \mathcal{B}$ after visiting cylinder L, that is, $\langle x \rangle \in \mathcal{B}_{(1)}$. As a result, the existence of the request $\langle q \rangle \in \mathcal{B}_{(1)} \cup \{\langle L' \rangle\}$ is clear. Notice that if $q = L$, then ADVB stays on cylinder L and services $\langle L^{W-1}q \rangle$, which equals $\langle L^W \rangle$.

Step P_3. ADVB simulates the remaining path of ADV, and replaces the requests in its delayed set visited during the move with appropriate requests such that the delayed set of ADVB, i.e., $\mathcal{B}_{(3)}$, matches $\mathcal{B}'$. Notice that the set of requests in $\mathcal{B}_{(2)}$ that are not visited during the move of ADVB is a subset of $\mathcal{B}'$, thus the construction in this step is feasible. This step fits Case VM.

The derivations of other general cases are similar to that in Case GC1. We list briefly their simulation steps in the following.

*Case GC*2

Step P_1. ADVB gives the request sequence $\langle A^W \rangle$. LOOK moves to A and services $\langle A^W \rangle$. ADV stays on A and services $\langle A^W \rangle$. (This step fits Case PC2.)

Step P_2. ADVB gives $\langle A^{W-1}L' \rangle$. LOOK moves to L' and service $\langle A^{W-1}L' \rangle$. ADVB services $\langle A^{W-1} \rangle$, then simulates the path of ADV until it reaches a cylinder q such that $\langle q \rangle \in \mathcal{B}_{(1)} \cup \{\langle L' \rangle\} = \mathcal{B} \cup \{\langle L' \rangle\}$, and then services $\langle q \rangle$. (Case PC1)

Step P_3. Identical to Step P_3 in Case GC1. (Case VM)

*Case GC*3

Step P_1. Similar to Step P_1 in Case GC1. ADVB simulates the path of ADV until it reaches L', and replaces each request in its delayed set visited during the move with a request on cylinder L'. (Case VM)

Step P_2. ADVB gives $\langle (L')^W \rangle$. (PC2)

Step P_3. Identical to Step P_3 in Case GC1. (VM)

*Case GC*4

Step P_1. Similar to Step P_1 in Case GC1. ADVB simulates the path of ADV until it reaches l, and replaces each request in its delayed set visited during the move with a request on cylinder l. (VM)
Step P_2. ADVB gives $\langle l^W \rangle$. (PC2)
Step P_3. ADVB gives $\langle l^{W-1} L' \rangle$. (PC1)
Step P_4. Identical to Step P_3 in Case GC1. (VM)

*Case GC*5. Let $\langle x_1 \rangle$ (resp. $\langle x_2 \rangle$) be the first (resp. second) request on the path of ADV that is in $\mathcal{B} \cup \{\langle l \rangle\} \cup \{\langle L' \rangle\}$. We consider the following three subcases:

Subcase 1: $x_1 = l$ *or* $\langle x_1 \rangle \in \mathcal{B}$ *and* $x_2 = l$.

Step P_1. Identical to Step P_1 in Case GC4. (VM)
Step P_2. ADVB gives $\langle l^W \rangle$. (PC2)
Step P_3. ADVB gives $\langle l^{W-1} L' \rangle$. (PC1)
Step P_4. Identical to Step P_3 in Case GC1. (VM)

Subcase 2: $x_1 = L'$ *or* $\langle x_1 \rangle \in \mathcal{B}$ *and* $x_2 = L'$.

Step P_1. Identical to Step P_1 in Case GC3. (VM)
Step P_2. ADVB gives $\langle (L')^{W-1} l \rangle$. (PC4)
Step P_3. Identical to Step P_3 in Case GC1. (VM)

Subcase 3: $\langle x_1 \rangle, \langle x_2 \rangle \in \mathcal{B}$.

Step P_1. Similar to Step P_1 in Case GC1. ADVB simulates the path of ADV until it reaches x_2, and replaces each request in its delayed set visited during the move with a request on cylinder x_2. (VM)
Step P_2. ADVB gives $\langle (x_2)^{W-2} l L' \rangle$. (PC3)
Step P_3. Identical to Step P_3 in Case GC1. (VM)

*Case GC*6

Step P_1. Identical to Step P_1 in Case GC3. (VM)
Step P_2. ADVB gives $\langle (L')^{W-1} l \rangle$. (PC4)
Step P_3. Identical to Step P_3 in Case GC1. (VM) □

5. Competitive Analysis of LOOK

In the following competitive analysis of LOOK, we use the technique of potential function, which has been employed in amortized analysis and competitive analysis. Let the request sequence under consideration be $\sigma = \langle r_1, r_2, \ldots, r_n \rangle = \langle \delta_1 \delta_2 \cdots \delta_p \rangle$, where δ_i, $1 \leq i \leq p = \lceil n/W \rceil$, is the ith phase of requests. Using the potential technique, we run LOOK and ADV phase by phase on σ, and compare the costs of LOOK and ADV phase by phase. The potential function maps a configuration of the disk at some time to a real number. Let Φ_{i-1} and Φ_i be the potentials at the beginning and at the end, respectively, of phase i, $2 \leq i \leq p-1$. (Notice that Φ_i is the potential at the end of phase i, and

is also the potential at the beginning of phase $i+1$.) We prove later in this section that $\text{cost}_{\mathsf{LOOK}}(\delta_i) + \Phi_i - \Phi_{i-1} \le \alpha \times \text{cost}_{\mathsf{ADV}}(\delta_i)$, for some α. If the potential is bounded from below and above, and the cost of LOOK in every phase is bounded, it is easy to derive that $\text{cost}_{\mathsf{LOOK}}(\sigma) \le \alpha \times \text{cost}_{\mathsf{ADV}}(\sigma) + \beta$, for some β. The cost of LOOK in the first and the last phase and $(\Phi_1 - \Phi_{p-1})$ can be grouped into β. That is, LOOK is α-competitive.

Let $(A, L, \mathcal{B})$ be the configuration of the disk. The potential Φ is defined as $M \times (\varphi_1 + \varphi_2 + \varphi_3 + \varphi_4)$, where

$$M = \frac{WT + D}{(2W-1)WT + D},$$

$$\varphi_1 = |A - L|,$$

$$\varphi_2 = -2 \times \sum_{\langle q \rangle \in \mathcal{B}} |A - q|,$$

$$\varphi_3 = \begin{cases} 2|A-L|, & \text{if} \quad (q-L) \times (L-A) > 0 \quad \text{for all} \quad \langle q \rangle \in \mathcal{B}, \\ 0, & \text{otherwise,} \end{cases}$$

$$\varphi_4 = K\left(\frac{1}{M} - 1\right), \qquad \text{where} \quad K = \min(L, D-L).$$

Intuitively, φ_1 is the distance between A and L. φ_2 is the potential charged on ADV when there are requests in the delayed set that are far from A. φ_3 is the potential charged on the special case where $A < L < q$, for all $q \in \mathcal{B}$, or $q < L < A$, for all $q \in \mathcal{B}$, to amortize the cost in the next phase where A moves closer to the requests in its delayed set. φ_4 is the potential that measures the distance from L to the near end of the disk. Notice that $-2M(W-1)D \le \Phi \le 3MD + \lfloor D/2 \rfloor (1-M)$, that is, the potential is bounded from below and above.

In the remainder of this section, we focus on the ith phase of operations of ADV and LOOK, $2 \le i \le p-1$, and, as in the previous section, we call this phase the current phase. Let C_A and C_L be the costs of ADV and LOOK, respectively, in the current phase. Let $(A, L, \mathcal{B})$, Φ, and $\varphi_i, 1 \le i \le 4$, be the configuration of the disk, the potential, and its four components, respectively, at the beginning of the current phase, and let $(A', L', \mathcal{B}')$, Φ', and φ_i' be the corresponding items at the end of the current phase. Let the potential difference in the current phase be $\Delta\Phi = \Phi' - \Phi$ and let $\Delta\varphi_i = \varphi_i' - \varphi_i$, $1 \le i \le 4$. We prove the following theorem about the competitiveness of LOOK.

Theorem 4. LOOK *is* $M(2W-1)$*-competitive. In particular, we have*

$$C_L + \Delta\Phi \le M(2W-1)C_A.$$

We first prove the theorem when $T = 0$ in Lemma 2 and then extend the result to the situation where $T \ge 0$ in Lemma 3. Notice that when $T = 0$, we have $M = 1$, $\varphi_4 = 0$, and $\Delta\Phi = \Delta\varphi_1 + \Delta\varphi_2 + \Delta\varphi_3$.

Lemma 2. LOOK *is* $(2W-1)$*-competitive when* $T = 0$. *In particular, we have*

$$C_L + \Delta\Phi \leq (2W-1)C_A. \tag{1}$$

Proof. First we show that (1) holds in the primitive cases and in the virtual moves of the adversary. At the end of the proof we show how to establish (1) for the current phase.

*Case PC*1. Notice that $A' \leq L'$. Thus, $C_A \geq |A'-A|$, $C_L = L'-A$, and $\Delta\varphi_1 = (L'-A')-0$. If $\langle L'\rangle$ is serviced by ADV, then $A' = L'$ and $\Delta\varphi_2 \leq 2(W-1)|A'-A|$ and $\varphi_3' = 0$; otherwise, one of the requests in $\mathcal{B}$ is serviced and $A' < L'$, then $\langle L'\rangle \in \mathcal{B}'$ and $\Delta\varphi_2 \leq 2(W-1)|A'-A| - 2(L'-A')$ and $\varphi_3' = 0$. In both situations we have $\Delta\varphi_2 \leq 2(W-1)|A'-A| - 2(L'-A')$ and $\Delta\varphi_3 \leq 0$. Therefore, $C_L + \Delta\Phi \leq (2W-2)|A'-A| + (A'-A) \leq (2W-1)|A'-A| \leq (2W-1)C_A$.

*Case PC*2. Notice that $L' = A' = A$ in this case. Thus $C_A = 0$, $C_L = (A-L)$, $\Delta\varphi_1 = 0-(A-L)$, and $\Delta\varphi_2 = 0$. Since $A' = L'$, we have $\varphi_3' = 0$ and $\Delta\varphi_3 \leq 0$. Therefore, $C_L + \Delta\Phi \leq 0 = (2W-1)C_A$.

*Case PC*3. In this case, $C_A = 0$, $C_L = L' + L - 2l$, $\Delta\varphi_1 = (L'-A) - |L-A|$, and $\Delta\varphi_2 = -2(L'-A) - 2(A-l)$. Since $\langle L'\rangle \in \mathcal{B}'$, we have $\varphi_3' = 0$ and $\Delta\varphi_3 \leq 0$. Therefore, $C_L + \Delta\Phi \leq (L-A) - |L-A| \leq 0 = (2W-1)C_A$.

*Case PC*4. Notice that $A' \geq l$. Let $x_{(+)}$ be the number of requests $\langle q\rangle \in \mathcal{B}$ such that $(A'-A)\times(q-A) \geq 0$, and let $x_{(-)}$ be the number of requests $\langle q\rangle \in \mathcal{B}$ such that $(A'-A)\times(q-A) < 0$. Notice that $x_{(+)} + x_{(-)} = |\mathcal{B}| = W-1$. In this case, $C_A \geq |A'-A|$, $C_L = A + L - 2l$, $\Delta\varphi_1 = |A'-A| - (A-L)$, and $\Delta\varphi_2 = 2x_{(+)}|A'-A| - 2x_{(-)}|A'-A| - 2(A'-l)$. Therefore, $C_L + \Delta\Phi = (2x_{(+)} - 2x_{(-)} + 1)|A'-A| + 2(L-A') + \varphi_3' - \varphi_3$. The discussion continues in the following four subcases:

Subcase 1: $x_{(-)} \geq 1$. It is easy to check that $(L-A') \leq |A'-A|$, $\varphi_3' \leq 2|A'-A|$, and $\varphi_3 \geq 0$. So $C_L + \Delta\Phi \leq (2W-1)|A'-A| \leq (2W-1)C_A$.

Subcase 2: $x_{(-)} = 0$ *and* $L \leq A' \leq A$. Since $\langle l\rangle \in \mathcal{B}'$ and $l < A' \leq L'$, $\varphi_3' = 0$. It is clear that $\varphi_3 \geq 0$ and $(L-A') \leq 0$. So $C_L + \Delta\Phi \leq (2W-1)|A'-A| \leq (2W-1)C_A$.

Subcase 3: $x_{(-)} = 0$ *and* $A' > A$. It is clear that $\varphi_3' \leq 2(A'-A)$ and $\varphi_3 \geq 0$. So $C_L + \Delta\Phi \leq (2W-1)|A'-A| + 2(L-A') + 2(A'-A) \leq (2W-1)|A'-A| \leq (2W-1)C_A$.

Subcase 4: $x_{(-)} = 0$ *and* $A' < L$. Since $q \leq A' < L$ for all $\langle q\rangle \in \mathcal{B}$, we have $\varphi_3 = 2(A-L)$ and $\varphi_3' = 0$. So $C_L + \Delta\Phi = (2W-1)|A'-A| + 2(L-A') - 2(A-L) \leq (2W-1)|A'-A| + 2(L-l) - 2(A-L) \leq (2W-1)|A'-A| \leq (2W-1)C_A$.

In the following we show that $\Delta\Phi \leq (2W-1)C_A$ in the virtual move. Notice that LOOK does not move during the virtual move, so $C_L = 0$.

Case VM. In this case, $\Delta\varphi_1 \leq |A'-A|$. Let k be the number of requests in the delayed set that are visited by ADV. Consider the following three subcases according to the values of $\Delta\varphi_3$ and k:

Subcase 1: $\Delta\varphi_3 \le 0$. In this subcase, $\Delta\varphi_2 \le 2(W-1)|A'-A|$ and $\Delta\Phi \le (2W-1)|A'-A| \le (2W-1)C_A$.

Subcase 2: $\Delta\varphi_3 > 0$ *and* $k = 0$. In this subcase, ADV moves away from all the requests in $\mathcal{B}$. Thus, $\Delta\varphi_2 = -2(W-1)|A'-A|$. It is clear that $\varphi_3' \le 2|A'-A|$ and $\varphi_3 \ge 0$. So, $\Delta\Phi \le (-2W+5)|A'-A| \le (2W-1)C_A$.

Subcase 3: $\Delta\varphi_3 > 0$ *and* $k > 0$. Assume that $\Delta\varphi_3 = 2d > 0$. Since the distance from the newly added requests to A' is at least d (otherwise, φ_3' will be 0), $\Delta\varphi_2 \le 2(W-1)|A'-A| - 2kd$. Therefore, $\Delta\Phi \le (2W-1)|A'-A| \le (2W-1)C_A$.

Let the cost of LOOK, the cost ADV, and the potential difference in step P_i in Lemma 1 be $C_{L(i)}$, $C_{A(i)}$, and $\Delta\Phi_{(i)}$, respectively. By Lemma 1 and the fact that (1) holds in the primitive cases and in the virtual moves of the adversary proved above, it is clear that $C_L + \Delta\Phi = \sum_{i=1}^{v}(C_{L(i)} + \Delta\Phi_{(i)}) \le (2W-1)\sum_{i=1}^{v} C_{A(i)} = (2W-1)C_A$. □

In the following we show how to extend the result of Lemma 2 to $T = t \ge 0$. For ease of expression, let $C_A^{T=t}$, $C_L^{T=t}$, and $\Delta\Phi^{T=t}$ be the values of C_A, C_L, and $\Delta\Phi$, respectively, when $T = t$.

Lemma 3. *If* $C_L^{T=0} + \Delta\Phi^{T=0} \le (2W-1)C_A^{T=0}$, *then* $C_L^{T=t} + \Delta\Phi^{T=t} \le m(2W-1)C_A^{T=t}$, *where* $t \ge 0$ *and* m *is the value of* M *when* $T = t$.

Proof. Let ΔK be the difference of K in φ_4 after and before the current phase. We show in the following that $\Delta K + C_L^{T=0} \le D$ in the current phase. Notice that we consider only cases where $L \le L'$. Those cases where $L' \le L$ are symmetrical. Let $u = \lfloor D/2 \rfloor$ denote the middle cylinder of the disk. We assume in the following cases that LOOK moves left $x \ge 0$ cylinders and then sweeps to L'.

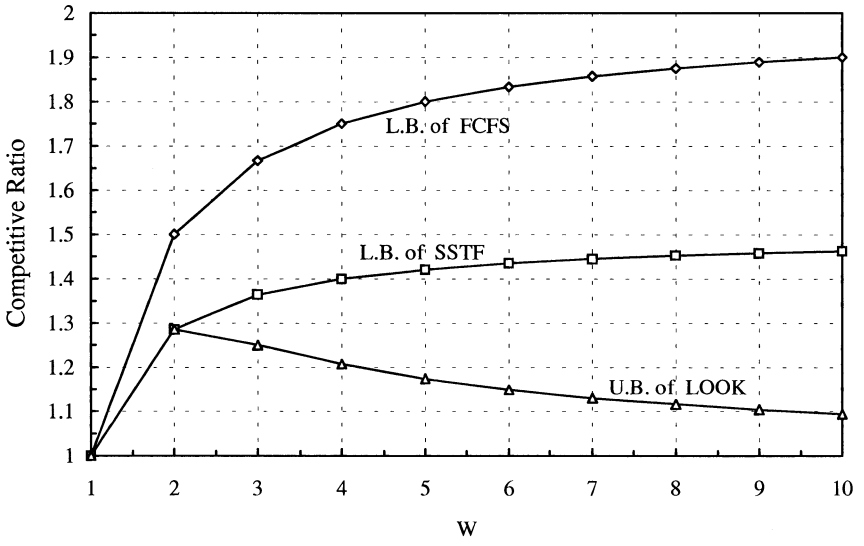
$L \le L' \le u$: Notice that $x + L' - L \le \lfloor D/2 \rfloor$. In this case, $\Delta K = L' - L$ and $C_L^{T=0} = 2x + L' - L$. Thus, $\Delta K + C_L^{T=0} = 2(x + L' - L) \le D$.

$L \le u < L'$: Notice that $x \le (L' - L)$ and $x \le L$. In this case, $\Delta K = (D - L') - L$ and $C_L^{T=0} = 2x + L' - L$. Thus, $\Delta K + C_L^{T=0} = D + 2x - 2L \le D$.

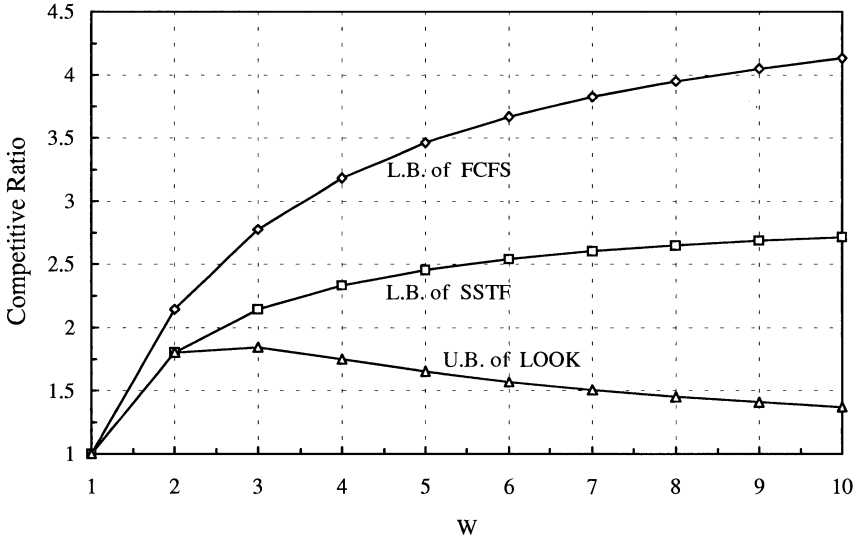
$u < L < L'$: Notice that LOOK may move to the left half of the disk after moving left x cylinders, and that $x \le (L' - L) \le \lfloor D/2 \rfloor$. In this case, $\Delta K = L - L'$ and $C_L^{T=0} = 2x + L' - L$. Thus, $\Delta K + C_L^{T=0} = 2x \le D$.

As a result,

$$
\begin{aligned}
C_L^{T=t} + \Delta\Phi^{T=t} &= C_L^{T=0} + Wt + m\left(\Delta\Phi^{T=0} + \Delta K\left(\frac{1}{m} - 1\right)\right) \\
&= m(C_L^{T=0} + \Delta\Phi^{T=0}) + m(2W-1)Wt + (1-m)C_L^{T=0} \\
&\quad + \Delta K(1-m) + Wt(1 - m(2W-1)) \\
&\le m(2W-1)C_A^{T=0} + m(2W-1)Wt + (1-m)C_L^{T=0} \\
&\quad + \Delta K(1-m) + Wt(1 - m(2W-1))
\end{aligned}
$$



(a) $D/T = 1$



(b) $D/T = 4$

Fig. 1. Comparison of FCFS, SSTF, and LOOK.

$$= m(2W-1)C_A^{T=t} + \frac{2(W-1)Wt}{(2W-1)Wt+D} \times (\Delta K + C_L^{T=0} - D)$$
$$\leq m(2W-1)C_A^{T=t}. \qquad \square$$

Clearly, Theorem 4 follows immediately from Lemmas 2 and 3.

6. Discussions and Conclusions

In this paper we have studied three disk scheduling algorithms FCFS, SSTF, and LOOK using competitive analysis. The cost measure in our model takes the seek startup time, the latency time, and the transfer time into consideration and groups them into a constant T. We showed that LOOK is $(WT + D)/(WT + D/(2W - 1))$-competitive, where D is the number of cylinders in the disk less one, and W is the size of the window, which represents the waiting buffer of the disk. We also provided the lower bounds of the competitive ratios of FCFS, SSTF, and LOOK. The competitive ratio of LOOK matches its lower bound.

Different disk drives may have different numbers of cylinders and different values of T. The D/T ratios of the disk drives discussed in [6], [14], and [15] range from 1 to 4. We plot the competitive ratio versus the window size for $D/T = 1$ and 4 in Figure 1. One can see from the figure that the competitive ratio of LOOK is lower than the lower bounds of the competitive ratios of SSTF and FCFS. That is, in a competitive sense, the performance of LOOK is better than those of SSTF and FCFS. Since the window size in our analysis reflects the work load of a disk system, the results strongly suggest that a system programmer should consider in favor of LOOK than SSTF and FCFS as the disk scheduling policy when the work load of the disk system is heavy. Though we are not able to provide the problem lower bound for disk scheduling, it is clear from Figure 1 that the competitive ratio of LOOK is very low, and we expect that the problem lower bound is close to the competitive ratio of LOOK.

Acknowledgments

The authors thank the anonymous referee for comments and suggestions which improved the correctness as well as the presentation of this paper.

References

[1] Y. Azar, A. Broder, and A. Karlin. On-line load balancing. *Theoret. Comput. Sci.*, 130:73–84, 1994.
[2] Y. Azar, J. Naor, and R. Rom. The competitiveness of on-line assignments. *J. Algorithms*, 18:221–237, 1995.
[3] T. Chen, W. Yang, and R. Lee. Amortized analysis of some disk scheduling algorithms: SSTF, SCAN, and *n*-step SCAN. *BIT*, 32:546–558, 1992.
[4] E. Coffman, Jr., and M. Hofri. On the expected performance of scanning disks. *SIAM J. Comput.*, 11:60–70, 1982.

[5] A. Fiat, R. Karp, M. Luby, L. McGeoch, D. Sleator, and N. Young. Competitive paging algorithms. *J. Algorithms*, 12:685–699, 1991.
[6] R. Geist and S. Daniel. A continuum of disk scheduling algorithms. *ACM Trans. Comput. Systems*, 5:77–92, 1987.
[7] M. Hofri. Disk scheduling: FCFS vs. SSTF revisited. *Comm. ACM*, 23:645–653, 1980.
[8] S. Irani. Coloring inductive graphs on-line. *Algorithmica*, 11:53–72, 1994.
[9] B. Kalyanasundaram and K. Pruhs. Online weighted matching. *J. Algorithms*, 14:478–488, 1993.
[10] R. Shimha and A. Majumdar. On lookahead in the list update problem. *Inform. Process. Lett.*, 50:105–110, 1994.
[11] D. Shmoys, J. Wein, and D. Williamson. Scheduling parallel machines on-line. *SIAM J. Comput.*, 24:1313–1331, 1995.
[12] A. Silberschatz, J. Peterson, and P. Galvin. *Operating System Concepts*, 3rd edn. Addison-Wesley, Reading, MA, 1991.
[13] D. Sleator and R. Tarjan. Amortized efficiency of list update and paging rules. *Comm. ACM*, 28:202–208, 1985.
[14] T. Teorey and T. Pinkerton. A comparative analysis of disk scheduling policies. *Comm. ACM*, 15:177–184, 1972.
[15] N. Wilhelm. An anomaly in disk scheduling: a comparison of FCFS and SSTF seek scheduling using an empirical model for disk accesses. *Comm. ACM*, 19:13–17, 1976.

Received February 1997, *and in revised form November* 1997, *and in final form February* 1998.