New Algorithms for the Disk Scheduling Problem *

Michael A. Bender† Matthew Andrews‡ Lisa Zhang§

The problem. Processor speed and memory capacity are increasing several times faster than disk speed which suggests that disk I/O performance may become an important bottleneck. Methods are needed for using disks more efficiently. Past analysis of disk scheduling algorithms has largely been experimental. Little attempt has been made to develop algorithms with provable performance guarantees or to determine the computational complexity of the problem. In this abstract we propose several disk-scheduling algorithms with performance guarantees and we state a hardness result.

We chiefly consider the off-line version of the disk scheduling problem. A computer disk rotates at a constant rate, and a movable *disk head* travels in and out radially in order to access locations on the disk. An instance of the off-line problem consists of a set of n locations on the disk. These n locations represent *requests* to be serviced by the disk head. To *service* a request, the disk head must be at the request and have no radial movement. Our goal is to schedule the disk head so that it services all of the requests in the smallest number of rotations possible. We also consider an on-line version of the problem in which the requests arrive over time and are placed into a queue. The head is able to service any request that is currently in the queue. Our goal is to maximize the throughput.

Between servicing two requests, the radial movement of the disk head consists of an acceleration phase, a full-speed phase (possibly of length zero) and a deceleration phase. In particular, this radial movement is determined by a *reachability function* $f(\theta)$. In a rotation of angle θ, the function $f(\theta)$ represents the maximum radial distance the head can travel when it starts and ends with no radial movement. The reachability function f is nondecreasing and convex.

We view the disk as a $2\pi \times 1$ rectangle, where the entire radial distance of the disk is normalized to 1. Each request R is specified by coordinates (θ, r), where $0 \leq \theta < 2\pi$ and $0 \leq r \leq 1$. The distance between request R and R' is defined to be the minimum number of times the disk head has to cross the line $\theta = 0$ between servicing R and R'. Note that this distance metric is integral and asymmetric. Let t_{fullseek} be the full seek time, i.e. the minimum number of rotations required for the head to travel the entire radial distance. We shall examine the motion of the head relative to the disk and so the head not only moves radially but also moves around the disk at a constant rate.

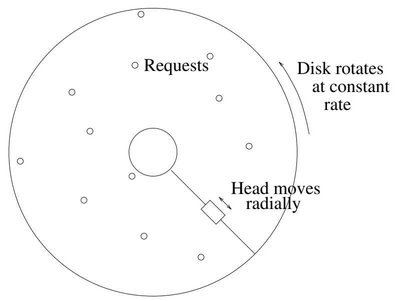

Figure 1: A computer disk.

An optimally solvable special case. We first consider the special case in which the reachability function is linear, i.e. the disk head either has no radial movement or has full radial speed. The linearity of f ensures the *reachability property*, i.e. regardless of its current speed, the head can follow any head path passing through its current position. If the disk head must start at location $(0, 0)$ of the disk and finish at $(0, 1)$, the reachability property implies that we can "uncross" the head trace of any solution. That is, any solution that requires m rotations can be transformed to a solution that has a head trace with no crossings and that also requires m rotations. (Note that a general reachability function does not have the reachability property. Hence the head trace cannot always be uncrossed.) We present an optimal polynomial-time algorithm which uses a greedy approach. The next request to be serviced is chosen to be the closest request to the current head position that can be reached without violating the no-crossing property.

Theorem 1 *If the reachability function is linear and the disk head starts at position $(0, 0)$ and ends at $(0, 1)$, then there exists an optimal polynomial-time algorithm.*

*This research is supervised by Professor David Karger and Professor Margo Seltzer. For an extended version see http://theory.lcs.mit.edu/˜andrews.

†Aiken Computation Laboratory, Harvard University. Supported by NSF contract 9313775-CCR. Advisor: Professor Michael Rabin. Email: bender@das.harvard.edu.

‡MIT Laboratory for Computer Science and Department of Mathematics. Supported by NSF contract 9302476-CCR, ARMY grant DAAH04-95-1-0607 and ARPA contract N00014-95-1-1246. Advisor: Professor Michel Goemans. Email: andrews@math.mit.edu.

§MIT Laboratory for Computer Science and Department of Mathematics. Supported by an NSF graduate fellowship, ARMY grant DAAH04-95-1-0607 and ARPA contract N00014-95-1-1246. Advisor: Professor Tom Leighton. Email: ylz@math.mit.edu.

NP-hardness and approximation algorithms We now consider the general problem. For convenience we enforce that the disk head must start and end at the same place. Using a reduction from an NP-hard special case of the directed Hamiltonian cycle problem in which each vertex is incident to 3 arcs, we show that the off-line disk scheduling problem is NP-hard.

Theorem 2 *For a given set of requests on the disk, it is NP-hard to find the optimal solution for general convex reachability functions.*

It is therefore not practical to look for an optimal general solution. We present two algorithms, one with a 3/2 approximation ratio (and a constant additive term) and the other with a $t_{\text{fullseek}} + 1$ approximation ratio (and no additive term). For modern disks, t_{fullseek} is usually at most 2. We study a directed disk graph, D, induced by the disk scheduling problem. In D the vertices are requests, edges are pairs of requests and the weight of a directed edge RR' is the distance from R to R'.

For the 3/2-approximation algorithm we first find a minimum-cost cycle cover[1] $\mathcal{C}$ of D in polynomial time using the Hungarian method. Let $C^{(i)}$ denote the set of cycles in $\mathcal{C}$ with cost i, then $\mathcal{C} = \cup_i C^{(i)}$ and $K = \sum_i i|C^{(i)}|$ is the is the cost of $\mathcal{C}$, where $|C^{(i)}|$ is the number of cycles in $C^{(i)}$. Note that K is a lower bound on the number of rotations required to service all the requests. We proceed to determine a particular order in which the cycles are serviced. Cycles in $C^{(1)}$ are serviced first, followed by cycles in $C^{(2)}$, $C^{(3)}$, etc. For each cycle we identify the first request to service and then travel around the cycle to service other requests. Within each $C^{(i)}$ the cycles are ordered such that it takes an extra $i/2$ rotations for the disk head to travel from the last request of a cycle to the first request of the next cycle for most cases. The additive term is to cover the cost of traveling between cycles in $C^{(i)}$ and $C^{(i+1)}$ and "a few" connections between cycles in $C^{(i)}$. Therefore,

Theorem 3 *There exists a polynomial-time algorithm which services all n requests in $\frac{3}{2}K + c$ rotations, where c is a term dependent on the reachability function but independent of n. Hence, the disk scheduling problem has a $\frac{3}{2}$-approximation algorithm.*

For the $(t_{\text{fullseek}} + 1)$-approximation algorithm, the disk scheduling problem is viewed as a special case of the asymmetric traveling salesman problem with the triangle inequality (ATSP-Δ). The best known approximation ratio for a general ATSP-Δ problem is $\lceil \log_2 n \rceil$. By constructing a partial order on the requests and applying Dilworth's Theorem, we obtain,

Theorem 4 *If all the edges in a directed graph either have length 0 or else have length between α and β for some constants $0 < \alpha \leq \beta$, then there exists a β/α-approximation algorithm for the resulting* ATSP-Δ *problem.*

Since each edge length in graph D is a member of $\{0, 1, \ldots, t_{\text{fullseek}} + 1\}$, Theorem 4 implies,

Corollary 5 *The disk scheduling problem has a $(t_{\text{fullseek}} + 1)$-approximation algorithm, where t_{fullseek} is the number of rotations for a full seek. (Usually t_{fullseek} is at most 2).*

The on-line problem In the on-line model of the disk scheduling problem, requests arrive over time and are placed into a queue. The disk head can only service requests that are in the queue. In real systems it is known [1] that requests often arrive in a "bursty" fashion, i.e. a large group of requests arrive together and then there is a period in which no requests arrive. We can use an off-line algorithm to service these requests. For experimental results using real trace data see [3].

We also consider the steady state model of disk arrivals used in [2]. The disk is discretized into tracks and sectors. A request must be a point that is on a track and at the beginning of a sector. Whenever a request is serviced it is replaced in the queue by a new request chosen *uniformly at random* from the allowable points on the disk. The *throughput* is defined to be the number of requests serviced divided by the number of sectors that have passed under the disk head. Seltzer et. al. [2] tested a number of algorithms experimentally. Of those considered, SHORTEST-TIME-FIRST (STF) had the highest throughput. STF always aims to service the request that can be reached with the smallest amount of rotation.

We propose an on-line algorithm CHAIN that is similar in spirit to the $(t_{\text{fullseek}} + 1)$-approximation algorithm. CHAIN aims to service the first request of the longest chain in the partially ordered set induced by the requests in the queue. Preliminary simulations indicate that the throughput of CHAIN is noticeably higher than that of STF.

[1] A *cycle cover* of a graph G is a collection of cycles in G such that every vertex of G is contained in exactly one cycle. The *cost* of a cycle cover is the sum of the weights of the edges in the cover.

References

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

[2] M. Seltzer, P. Chen, and J. Ousterhout. Disk scheduling revisited. In *USENIX*, pages 313–324, Winter 1990.

[3] B. L. Worthington, G. R. Ganger, and Y. N. Patt. Scheduling algorithms for modern disk drives. In *SIGMETRICS*, pages 241–251, 1994.