Analysis of the GSTF disk scheduling algorithm

Eitan Bachmat * Ilan Elhanan †

1. INTRODUCTION

Modern disk drives have the ability to queue incoming read and write requests and to service them in an out of order fashion. In the disk scheduling problem we are given at any given moment a set of queued requests and we wish to service them in an order which minimizes the service rate, or equivalently, in an order which maximizes the number of requests serviced per time unit.

The disk scheduling problem with various assumptions has a long history of both experimental and theoretical work, see [2, 3, 4, 5, 6, 8, 9, 12, 11] among many others.

In [12], Seltzer, Chen and Ousterhout suggest several algorithms for the disk scheduling problem in an attempt to achieve good performance without starving requests, i.e., that the maximal response time for a request remains low. At first they consider the greedy algorithm which chooses as the next request to be serviced, the request which can be satisfied in the least amount of time given the current disk head position. This policy is known as `SPT` (shortest positioning time). This policy has very good overall performance, but some neglected requests may wait a very long time to be serviced. To alleviate this problem the authors suggest that the disk drive be divided into several cylindrical regions, given by ranges of the radius of the request. The algorithm, known as `GSTF` (Grouped shortest time first) , services all accumulated requests to a given region of the disk drive (ignoring incoming requests to the region during the service time) and moves on to service the accumulated requests of the next region.

In this paper we will analyze the `GSTF` algorithm. The analysis of the basic case is closely related to the card games called Bulgarian solitaire and Austrian solitaire, see [1, 10].

*Department of Computer Science, Ben-Gurion University, Beer-Sheva, Israel. Email: ebachmat@cs.bgu.ac.il. Research supported in part by an Israeli Science Foundation research grant

†Department of mathematics, Ben-Gurion University, Beer-Sheva, Israel, Email:ilanel@post.bgu.ac.il

We show that if I/O requests are uniformly distributed, in the limit, as the number of regions becomes large, the `GSTF` algorithm has a throughput which is $\sqrt{2}$ times larger than in the case of a single region.

The `GSTF` algorithm implicitly assumes that the distribution of requests to the disk is uniformly random. We introduce a variant of `GSTF` which handles non uniform distributions better in terms of performance while somewhat increasing the risk of starvation. We provide a partial analysis of the variant. the variant seems to exhibit rather complex behavior.

2. PRELIMINARIES

We model the disk drive platter in polar-like coordinates. For convenience, the radial coordinate will have range $0 \leq r \leq 1$ and similarly, we let the angular coordinate have range $0 \leq \theta \leq 1$, instead of the more standard range in $[0, 2\pi]$. The annulus is formed via the identification of points $(r, 0)$ with $(r, 1)$. Note however, that unlike polar coordinates we do not identify the points $(0, \theta)$, since the platter is an annulus. We can conveniently normalize our time parameter t so that at time $t = 0$, the angular position of the head of the disk is $\theta = 0$, or in other words, $\theta(0) = 0$. We can also normalize the time parameter so that the disk platter makes a full rotation after precisely one time unit. Given that the platter rotates at a constant speed and that the head only moves radially, this implies that

$$\theta(t) = t \mod 1 \tag{1}$$

The **seek time function** $f(r)$ measures the time required for the head to radially move a distance r. The motion must start and end without radial velocity since the disk head must be stable to read and write data at a given radius.

The constant speed rotation of the disk which is responsible for equation (1), together with the seek time function provide the following mathematical characterization of disk drive performance.

DEFINITION 1. *Let $L_1 = (r_1, \theta_1)$ and $L_2 = (r_2, \theta_2)$ be two locations on a disk drive with seek time function f. We define $d_f(L_1, L_2)$, the time required to move from location L_1 to location L_2 as the minimum of $t_2 - t_1$ taken over all pairs t_1, t_2 such that $\theta(t_1) = \theta_1$, $\theta(t_2) = \theta_2$ and $t_2 - t_1 \geq f(|r_1 - r_2|)$.*

We will assume that the seek distance function f is linear and is normalized so that $f(t) = t$. Physically, this choice amounts to ignoring the effects of acceleration and deceleration of radial disk head motion. We assume that the

disk head reaches maximal velocity instantaneously and similarly, can stop instantaneously. Obviously, both assumptions are unrealistic, but it is easier to tackle this toy model and it provides us with some basic insights.

We assume that in any given point in time we have n I/O requests, whose locations are chosen in i.i.d. fashion from a distribution $\mu_p = p(r,\theta)drd\theta$. When an I/O request is handled by the disk drive, a new request is introduced to the queue. This way the number of requests in the queue remains constant. Due to commonly practiced data layout considerations, usually, the distribution $p = p(r)$ depends only on r. The distribution which was chosen in the simulation evaluation of the `GSTF` algorithm in [12] was the uniform distribution given by $p(r) = 1$.

In the case of a linear seek function and a given set of n given requests, Andrews, Bender and Zhang (ABZ) have found a simple algorithm for computing the optimal order of service, [2].

We describe a family of I/O scheduling algorithms which we call `GSTF` algorithms.

A `GSTF` algorithm consists of the choice of domains D_i, $i = 1, ..., k$ in the disk drive. The union of the domains covers the disk drive and they can only intersect along boundaries. At each stage of the algorithm a domain is chosen, and all the outstanding requests in the domain are serviced (ignoring newly arriving requests) using the ABZ algorithm. Once all the requests have been serviced another domain is chosen.

In the version of the `GSTF` algorithm presented in [12], the domain D_i was given by $\frac{i-1}{k} \leq r \leq \frac{i}{k}$. The choice of the domain consisted of the periodic sequence

$$D_1, D_2, ..., D_k, D_1, D_2, ..., D_k, D_1$$

We will call the `GSTF` algorithm with these choices a *standard* `GSTF` *algorithm*. We will say that a domain is radial if it is given by $a \leq r \leq b$.

We will measure performance by the service rate, i.e., the average time required to service a single request.

The following result, is an amalgamation of results from [3] and [7] and [13] provides an asymptotic formula for the service time of the ABZ algorithm on n requests sampled with respect to a radial distribution in a radial domain.

THEOREM 1. *Consider a disk drive with seek time function $f(t) = t$. Let D be a radial domain of the form $a \leq r \leq b$ in the cylinder C, representing disk locations. Let $p(r)drd\theta$ be a radial distribution on D. Let $\bar{R}$ be a set of n requests in D, sampled i.i.d. from the density p and let $ST(\bar{R})$ be the number of disk rotations needed to service all the requests in $\bar{R}$ with an optimal policy. Then: For all $\varepsilon > 0$, there is a $\delta = \delta(\varepsilon)$ such that with probability at least $1 - e^{\delta n}$*

$$|\frac{ST(\bar{R}) - diam(D)}{\sqrt{n}}| < \varepsilon\sqrt{n} \qquad (2)$$

where $diam(D) = \sqrt{2}\int_a^b \sqrt{p(r)}dr$.

We define the average service rate of a scheduling algorithm with m outstanding requests as $R_m = lim_{t\to\infty} S_m(t)/t$, where $S_m(t)$ denotes the number of requests serviced by time t. We will consider algorithms where the above limit exists (and is a given constant) with probability approaching 1, and study the asymptotic behavior of R_m as $m \to \infty$. In particular, we define the normalized asymptotic service rate to be $\tilde{R} = lim_{m\to\infty} R_m/\sqrt{m}$, assuming the limit exists. From the theorem above, we conclude that the GSTF algorithm with a single region D consisting of the whole disk and with uniformly dense I/O requests has a normalized rate of $\tilde{R} = 1/\sqrt{2}$

3. ANALYSIS OF THE GSTF ALGORITHM

We will prove the following result:

THEOREM 2. *For a uniform distribution of requests, linear seek function and k regions, the normalized rate of the standard* `GSTF` *algorithm is*

$$\tilde{R} = \sqrt{\frac{k}{k+1}}$$

Sketch of proof:

We need to estimate how many requests are in a disk region when we apply the algorithm to that region. The following discrete Markov chain describes the evolution of the number of requests in the domains D_i of the `GSTF` algorithm, applied to m outstanding requests.

denote by $X_m(n,i)$ the number of requests in the i-th region after n steps (a single step occurs every time we apply the ABZ algorithm to a region of the disk) when the algorithm is applied with m outstanding requests. Denote by $X_m(n)$ the k-tuple $(X_m(n,1), ..., X_m(n,k))$

The Markov chain evolves as follows. At time $n = i(mod\ k)$, we service all the $X_m(n,i)$ requests in the $i-th$ region, then we get $X_m(n,i)$ new requests, each one of the new requests is placed independently in a region which is chosen by sampling the uniform distribution on k elements. The resulting state is the n+1 state of the chain.

Denote by $Y_m(n)$ the process $\frac{X_m(nk)}{m}$. We wish to show that when m and n are large, $Y_m(n)$ is with high probability close to the vector L_k, with entries $L_k(i+1) = \frac{2(k-i)}{k(k+1)}$, $i = 0, ..., k-1$. We prove the following result:

THEOREM 3. *for each $\epsilon > 0$ there exist a time n_0 and a constant C such that for each time $n \geq n_0$:*

$$Pr(\|Y_m(n) - L(k)\|_\infty > \epsilon) \leq \frac{C}{m}$$

where Pr denotes the probability of an event.

The strategy to prove theorem 3 is by showing that the process $Y_m(n)$ converges to a deterministic process when m tends to infinity. The deterministic process converges to $L(k)$ when n tends to infinity. These two facts together will give us theorem 3.

Consider the following deterministic process $\tilde{X}(n)$:

There are are k regions numbered from 0 to k-1. Each region contains a fluid of some level $q_i \geq 0$, with $\sum_i q_i = 1$. The sum of the fluid in the system stays constant and is equal to 1. At time n=i(mod k) we redistribute the contents of the $i-th$ cell evenly among all cells (including the $i-th$ cell). The result is the n+1 stage of the process.

The following 2 lemmas make the idea in the above paragraphs concrete:

LEMMA 1. *For each $\epsilon > 0$ and each time n_0 there exists a constant C such that if $\frac{X_m(0)}{m} = \tilde{X}(0)$ and $0 \leq n \leq n_0$ than:*

$$Pr(\|\frac{X_m(n)}{m} - \tilde{X}(n)\|_\infty > \epsilon) \leq \frac{C}{m} \quad (3)$$

LEMMA 2. *Let $Y(n)$ be the process $\tilde{X}(nk)$, then, $Y(n)$ converges to $L(k)$. moreover, the convergence is uniform in the initial states, i.e., for each $\epsilon > 0$ there exist N such that for each $n \geq N$ and for each vector $\overrightarrow{V} \in \mathbb{R}^k$ such that $Y(0) = \overrightarrow{V}$, $\|Y(n) - L(k)\|_\infty \leq \epsilon$.*

We need to compute $\tilde{R} = lim_{m\to\infty} R_m/\sqrt{m}$. Theorem 3 essentially says that apart from an exponentially small portion of the steps, the number of requests we serve at each step is roughly $\frac{2m}{k+1}$ requests, and from this, a simple computation provides the normalized rate in the theorem.

4. A VARIANT OF THE GSTF ALGORITHM

We present a variant of the `GSTF` algorithm. The algorithm again consists of a choice of domains in the disk drive. The difference is that the domains are not serviced in a prescribed order. Instead, after a domain has been serviced, the next domain to be serviced is chosen to be the one containing the most requests. This variant may be more efficient, especially when the probability measure is far from being uniform.

To analyze this variant we need to know how many requests on average are in the region we are serving. To do this we need to explore a Markov chain which describes how many requests are in each region over time. This chain is superficially similar to the corresponding Markov chain describing the standard `GSTF` algorithm. However, there are 2 differences. The first one is we no longer assume the distribution is uniform, therefore each region D_i has a weight α_i associated with it. The weights are numbers between 0 and 1 such that their sum is 1. When we get the new requests instead of being placed into cells according to a uniform distribution, the requests are placed according to the weights α_i. The second difference is that the next region to be serviced will be the one with the most requests.

There is also a corresponding deterministic process in which a fluid is distributed among the regions. We examined some cases of 2 regions.

The deterministic process may be viewed as iterating the function f on the interval [0,1]. Where f is defined as follows:

$$f(r) = \begin{cases} \alpha r & \text{if } r > \frac{1}{2} \\ \alpha + (1-\alpha)r & \text{if } r \leq \frac{1}{2} \end{cases}$$

The behavior of the process depends on the parameter α. The following result provides an analysis of the dynamics of the deterministic process for a certain range of the parameter.

THEOREM 4. *when α satisfies the following 2 equations:*

$$1 - \alpha^{n-1} - \alpha^n \geq 0$$

$$1 - 3\alpha^{n-1} + \alpha^n < 0$$

There exist a unique, globally attracting periodic orbit of length n, i.e., no mater what the starting point is, the process converges to that periodic orbit.

The conditions on α in the theorem define an infinite set of semi-open intervals with mutually exclusive closures. We also know that for some other values of α there are more complicated periodic orbits. In ongoing work we are trying to analyze the dependence of the dynamics of the map f on the parameter α, we expect rather complex dependence.

Additionally, there are many interesting open questions on the behavior of `GSTF`, such as monotonicity of performance with respect to partitioning of regions.

Acknowledgments: E. Bachmat and I. Elhanan have been supported by grant 580/11 of the Israeli Science Foundation.

5. REFERENCES

[1] E. Akin and M. Davis, Bulgarian solitaire, *American Mathematical Monthly*, 92(4), 237-250, 1985.

[2] M. Andrews, M.A. Bender, and L. Zhang, New algorithms for the disk scheduling problem, *Algorithmica*, 32, 277-301, 2002. Conference version, *Proceedings of FOCS*, pages 580-589, October 1996.

[3] E. Bachmat, Average Case Analysis of Disk Scheduling, Increasing Subsequences and Spacetime Geometry, *Algorithmica*, 49(3), 212-231, 2007.

[4] F. Cady, Y. Zhuang, and M. Harchol-Balter. A Stochastic Analysis of Hard Disk Drives, *International Journal of Stochastic Analysis*, Article ID 390548, 2011.

[5] E.G. Coffman, L.A. Klimko and B.Ryan, Analysis of scanning policies for reducing disk seek times, *SIAM Journal of computing*, 1(3), 1972.

[6] P.J. Denning, effects of scheduling on file memory operations, *Proceedings of AFIPS spring joint computer conference*, 9-21, 1967.

[7] J.D. Deuschel and O. Zeituni, On Increasing Subsequences of I.I.D. Samples, *Combinatorics, Probability and Computing*, 8, 247-263, 1999.

[8] H. Frank, Analysis and optimization of disk storage devices for time-sharing systems, *Journal of the ACM* 16(4), 602-620, 1969.

[9] C.C. Gotlieb and G.H. MacEwen, Performance of movable-head disk storage devices, *Journal of the ACM*, 20(4), 604623, 1973.

[10] S. Popov, Random Bulgarian solitaire, *Random Structures and Algorithms*, 27(3), 310-330, 2005.

[11] A. Riska, E. Riedel and S. Iren, Adaptive disk scheduling for overload management , *in Proceedings of the 1st International Conference on Quantitative Evaluation of Systems, (QEST)*, 176-186, 2004.

[12] M. Seltzer, P. Chen, and J. Ousterhout, Disk scheduling revisited, *in Proceedings of the USENIX Technical Conference*, 313324, 1990.

[13] T. Seppalainen, Large deviations for increasing sequences on the plane, *Probability Theory and Related Fields*, 112, Issue 2, 221-244, 1998.