Erciyes Üniversitesi Fen Bilimleri Enstitüsü Dergisi **22**(1-2) 11-19 (2006)
http://fbe.erciyes.edu.tr/
ISSN 1012-2354

OPTIMAL DISK SCHEDULING BASED ON ANT COLONY OPTIMIZATION ALGORITHM

S. ÖKDEM, D. KARABOĞA*

Department of Computer Engineering, Erciyes University, Kayseri, TURKEY

ABSTRACT

Disk scheduling problem has theoretical interest and practical importance since, the processor speed and memory capacity have been progressing several times faster than disk speed. More efficient disk usage methods have been needed because of the slow evolution in disk speed technology. Although disk and memory capacity increment is much higher versus disk speed improvement, little studies have been made to develop disk usage algorithms in more efficient manner. In this work, a new approach based on ant colony algorithm is proposed for disk scheduling problem and its performance is evaluated.

Keywords: Disk scheduling, Ant colony optimization.

KARINCA KOLONİ ALGORİTMASINA DAYALI OPTİMAL DİSK PLANLAMASI

ÖZET

Disk hızına nazaran işlemci hızı ve hafıza kapasitesinin kat kat hızlı gelişim içerisinde olmasından dolayı disk planlama problemi teorik ilgi ve pratik öneme sahip olmuştur. Disk hızlandırma teknolojisindeki yavaş ilerleme nedeniyle daha etkin bir şekilde disk kullanımı metotlarına gerek duyulmaktadır. Disk ve hafıza kapasitesi gelişimi disk hızı gelişiminden çok daha üst seviyede olmasına rağmen, daha verimli bir şekilde disk kullanımını sağlayan algoritmaların geliştirilmesi için az sayıda çalışma yapılmıştır. Bu çalışmada, disk planlama problemi için karınca koloni algoritması üzerine kurulu yeni bir yaklaşım önerilmiştir ve bu yaklaşımın performansı değerlendirilmiştir.

Anahtar kelimeler:

* **E-posta:** karaboga@erciyes.edu.tr

1. INTRODUCTION

In a modern computer, the performance of overall system is directly affected by processor speed, memory and disc capacity and disc speed. Although processor speed and memory capacity are increasing by over 40% per year, evolution of disc speed increases more gradually, growing by only 7% per year [1]. The reason of lower progress of disk speed technology is mainly related to difficulty of improvement of mechanical components. However, performance improvements of disc operation, transferring data at higher bandwidths, can still be provided by developing more efficient disc scheduling algorithms.

A disk drive contains a set of rotating plates having many circular tracks on. Tracks holding a specific data may be in sequential order or separated from each other. The head of disk moves between tracks to collect data over disk plate. When accessing tracks, the head suffers from mainly two kinds of delay: *seek time* required to reach the desired track and *rotational latency* for the requested sector to pass underneath the head.

Several scheduling algorithms have been proposed to achieve higher performance by taking into account some information about disk requests (parameters). The *seek time* is the main parameter or usually the only parameter used in the disk scheduling algorithms. Because *rotational time* is much lower than *seek time*, it is omitted in most algorithms. The other main parameters are request priority, request deadline and the request type [2, 3]. One of the most popular disc scheduling algorithm CSCAN [4], is used in the UNIX operating system and tries to reduce the seek time using only one parameter. The head, in this algorithm, travels from the outermost to the innermost track, servicing all the requests as moving over the tracks. Although this algorithm considered useful in the past, as the type of data differs, where priorities and deadlines involve, CSCAN algorithm remains less effective because of discarding other parameters except from *seek time*.

As computer systems developed, the usage of its components is also complicated. In real-time systems, *deadline* is an essential parameter for the requests of operations. In the client/server applications, the *priorities* of the tasks are inevitable parameters to proceed operations regularly. Therefore, also in disc requests, multiple parameters and criteria should be considered in disc scheduling jobs. A novel algorithm proposed in this paper, supplies a feasible solution to be easily adapted to disc scheduling problems in real-time and/or client server environment. The algorithm uses ant colony optimization to move the disc head properly to the locations required by the requests, a special case of traveling salesman problem, while considering *priorities* and *deadlines*. Disc scheduling is a significant factor on the disk performance where it is especially important in such systems as database servers, web servers, or mail servers [5]. Recent developments in field gate programming technology [6] give opportunities to embed heuristic algorithms in real-time environments for problems such as disc scheduling. For example, current implementations of ant colony optimization, which is a popular heuristic, in these environments have been simple structured allowing running of ant colony algorithms having small populations [7].

In this work, a disc scheduling algorithm based on ant colony optimization has been proposed. With the technological advances in real-time field gate programming techniques, the proposed disc scheduling algorithm can also be embedded in real-time devices to be used in disc devices easily. The disk scheduling problem is described in section 2. The well known conventional algorithms used for disk scheduling policy are FCFS (First Come First Served), SSTF (The Shortest Seek Time First), SCAN, LOOK, C-SCAN (Circular SCAN), C-LOOK (Circular LOOK), EDF (Earliest Deadline First), SCAN-EDF, DSDRP (Disk Scheduling with Dynamic Request Priorities), WSTTF (Weighted Shortest Total Time First). These algorithms are presented in section 3. A new approach based on ant colony algorithm is described in section 4 and the simulation results are presented in section 5.

2. DISK SCHEDULING PROBLEM

A disk consists of a disk head and a number of cylinders. A cylinder has a set of tracks having same distances from the disk center. Each track has sectors and each sector has 32 bytes. The disk request < r > is on the cylinder *r*.

The distance between cylinders r_1 and r_2 is $| r_1\text{-}r_2 |$. A request sequence is the disk requests waiting to be serviced, $\alpha =< r_1, r_2, \ldots r_n >$. Size of the request queue *n* is the value depending on application [2].

An example disk model with a single data disk using a single head has the parameters shown in Table 1.

Table 1. The parameters of the disk device.

Parameter	Meaning	Value
Tracks	Number of Tracks on Disk	1000
Seek Factor	Seek Time Scaling Factor	0,6ms
Disk Constant	Rotational Latency + Transfer time	15,0ms

The access time for a disk request having n tracks distance from the current head position is given by equation 1.

$$Access_n = Seek_n + Rotational\ Latency + Transfer\ Time \quad (1)$$

In this model, rotational latency and transfer time are grouped in single parameter, Disk Constant; and seek time, $Seek_n$, can be expressed in a nonlinear equation depending on seek factor and track number.

$$Access_n = SeekFactor \times \sqrt{n} + Disk\ Constant \quad (2)$$

For disks having large number of tracks, seek time have more effect on the equation 2 and disk constant term can be omitted. Therefore, most of disk scheduling algorithms consider only track number n as the main parameter.

Client/Server distributed systems must manage disk I/O resources efficiently and fairly between multiple clients having requests varying urgencies. Therefore, the parameter request priority should be used in these systems [4].

Real-time applications require that their disk I/O requests be serviced within a bounded time. To provide real time requests not to be missed, the constraint considering request deadlines should be taken into account. In real-time systems request deadline is an important parameter [10].

Hence, one or a group of parameters are used in the cost function of a disk scheduling algorithm. The algorithm determines which order of requests produces the best performance. The performance can be evaluated using the following criterions: throughput which is the number of completed requests in a specific time; response time which is the time slice between entering into request queue and beginning to be served; waiting time which is the time slice between entering into request queue and the completion of request service; starvation which is a situation for a request waiting for a long time to be served and overhead which is the total time elapsed for the scheduling algorithm to perform the request order.

3. CONVENTIONAL DISK SCHEDULING TECHNIQUES

Conventional techniques are mainly categorized in two groups. The first group of traditional scheduling algorithms does seek optimization with the use of track information only. The second group of algorithms makes the use of additional information such as request deadline or priority. Some well-known algorithms existing in the first group are briefly explained below:

FCFS: The First Come First Served algorithm is preferable when disk load is light. In fact there is no scheduling task, requests are served in order of time entering into queue. This algorithm has the smallest overhead but FCFS is not successful in terms of other criterions.

SSTF: The Shortest Seek Time First policy improves throughput by servicing the request that results in the shortest seek distance. This algorithm provides high throughput but causes starvation problem.

SCAN: This algorithm is also named *elevator* algorithm which provides a lower response time variance (less starvation) than SSTF. The disk arm shuttles back and forth across the entire range of cylinders, servicing all requests in its path. It must be considered that the center requests are favored about two times more than edge requests.

C-SCAN: Circular SCAN algorithm is a derivation of SCAN algorithm which replaces the bidirectional scan with a single direction of arm travel. When the arm reaches the last cylinder, a full stroke seek returns it to the first cylinder without servicing any requests along the way. Thus, the variance of response time between center and edge requests is minimized.

LOOK: This algorithm is a variant of SCAN where the disk arm does not reach to innermost or outermost. The arm switches its direction if there is no more outstanding request in that direction and works as in SCAN algorithm.

C-LOOK: Circular LOOK is a variance of LOOK algorithm. Like C-SCAN algorithm, it tries to decrease variance of response time, servicing only one direction [11].

Some algorithms employing two parameters are explained below:

EDF: The Earliest Deadline First algorithm is an analog of FCFS having zero affect of track number parameter. Disk requests are ordered according to deadlines and the request with the earliest deadline is served first.

SCAN-EDF: This strategy is a combination of the SCAN and EDF. The request with the earliest deadline is always served first. If the requests have the same deadline, the specific one that is first according to SCAN direction is served first [4,5].

DSDRP: Disk Scheduling with Dynamic Request Priorities algorithm is a scheduling policy in client/server systems. The SCAN algorithm is modified to support prioritized requests. Disk requests are grouped on the basis of their priority, and within each group SCAN algorithm is used. The lower priority groups are serviced only when higher priority groups are empty [11].

WSTTF: Weighted Shortest Total Time First is an algorithm based on standard SSTF, but an aging function is used. In this function a request chance to be served is increased by its waiting time value. So this policy provides less starvation while keeping high throughput [12].

4. A NEW APPROACH BASED ON ANT COLONY ALGORITHM

Real ants can find the shortest way from a food source to their nest. While finding the way, they do not use visual cues. If the shortest way changes due to environmental conditions, they are able to find the new shortest way in an amount of time. An ant deposits a chemical substance named pheromone on its way while walking and probabilistically prefers to follow a direction rich in pheromone. Ants choosing the shorter path more rapidly reconstitute the pheromone compared with those choosing the longer path. Thus, the shorter path receives a great amount of pheromone per time unit and, in turn, a larger number of ants choose the shorter path. Because of this positive feedback, most of ants rapidly choose the shorter path. The behavior of ants explained above can be simulated to solve particularly traveling-salesman type optimization problems. Ant Colony Optimization (ACO) algorithms simulate this behavior of real ants. In ACO several generations of artificial ants search for good solutions to the problem.

The disk scheduling aims to find the best order of disk requests considering the specific request parameters such as track number, priority, deadline etc. The disk scheduling problem is similar to traveling-salesman problem. In the

analogy, the shortest food path of ants can represent the optimum order of requests in request queue. In this work, ACO algorithm is used for optimally disk scheduling in a way of modified traveling salesman problem.

Each ant has a memory, M_k, to record disk positions already visited. The memory is emptied at the beginning of each new tour, and updated after selecting next disk position. The artificial ant k in the present disk position r chooses the next disk position s to move to among those which is not in memory M_k by applying the following probabilistic formula:

$$P_k(r,s) = \begin{cases} \dfrac{\tau(r,s)^{\alpha}\,\eta(r,s)^{\beta}}{\sum_{s \notin M_k} \tau(r,s)^{\alpha}\,\eta(r,s)^{\beta}} & \text{if } s \notin M_k \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

where $\tau(r,s)$ is the amount of pheromone on the edge (r,s), $\eta(r,s)$ is a heuristic function (the inverse of the distance between tracks), α and β are the parameters that control the relative importance of pheromone versus heuristic, $P_k(r,s)$ is the probability with which ant k chooses to move from the present position r to the position s.

At the beginning ants start from the current disk head position. After the ant k completes the selection process of all disk positions, the tour length, $TourLength_k$, is calculated by the summation of distances between tracks visited. The amount of pheromone $\Delta\tau_k(r,s)$ deposited on each visited edge by the ant k is computed by the following formula:

$$\Delta\tau_k(r,s) = (TourLength_k)^{-1} \tag{4}$$

After all ants in the colony produce their tours, the pheromone of edge (r,s) is modified by means of Eq.5.

$$\tau(r,s) = \sum_{k=1}^{n} \Delta\tau_k(r,s) + \alpha\tau_0 \tag{5}$$

where n is the number of ants preferring the edge (r,s) to form the tour, τ_0 is the previous pheromone amount of the edge (r,s) and α is the evaporation parameter [10].

5. SIMULATION RESULTS

The simulation was built to model the disk requests with a single data disk, which has 840 cylinders, in which the equation of disk specifications is given in Table 2 [9]. The performance of new algorithm is tested with total response time criterion with a single parameter track number and multiple parameters track number, priority and deadline. Since the rotational latency is much lower than seek time, it was neglected in this study.

Table 2. Specification of the disk model.

$\boldsymbol{ResponseTime_x}$	
0	if x = 0
$4.6\times10^{-3} + \sqrt{x}\times0.87\times10^{-3}$ sec	if x ≤ 239
$18\times10^{-3} + (x-239)\times0.028\times10^{-3}$ sec	if x > 239

In Table 2, *x* represents the track number between the present position and the next position to be reached.

In the application, firstly, the parameter values related to the problem must be specified. Main parameters are queue size which is the maximum number of disk requests to be hold in the queue, maximum track number of the disk device. Secondly, the values of control parameters of ACO must be determined. Algorithms were tested with the queue size of 10, 20 and 50 respectively.

Since ACO is a probabilistic algorithm, it might produce a different solution for each run. Therefore, ACO was executed ten times for each case to calculate its average performance. In this study, the maximum iteration number and the ant number in the colony for 10 and 20 queue size were selected to be 50 and for larger queue size (50), these parameters were selected to be 75. The evaporation parameter was set to 0,8. These values were obtained where the variance of different ACO algorithm runs was not to exceed 5 percent.

The results given in Table 3-8 present the calculated average response times found by the algorithms in terms of different queue sizes and parameters. Algorithm names are given below the tables. The presented response times in the tables were calculated by means of equations in Table 2.

It is seen from the tables that the performance of ACO is much better than that of other algorithms in terms of response time.

6. CONCLUSION

In this work, a new approach based on ant colony optimization algorithm was proposed to solve disk scheduling problem. The performances of the proposed and the traditional approaches were compared. From the simulation results it was observed that the proposed approach produces superior performance.

ACKNOWLEDMENTS

This work is supported in part by Erciyes University Scientific Research Projects Fund (EUBAP FBA-04-13).

Table 3. Average of algorithm responses for queue size 10 with single parameter.

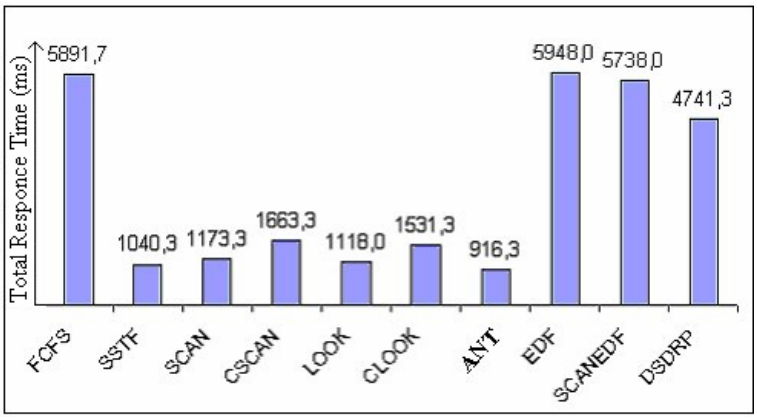


Table 4. Average of algorithm responses for queue size 20 with single parameter.

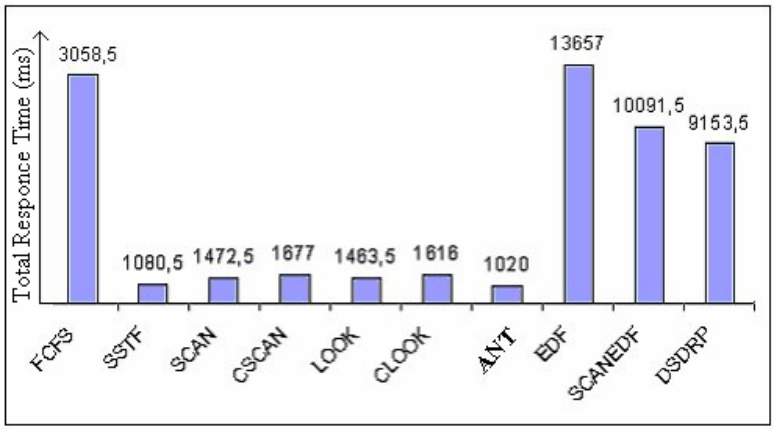


Table 5. Average of algorithm responses for queue size 50 with single parameter.

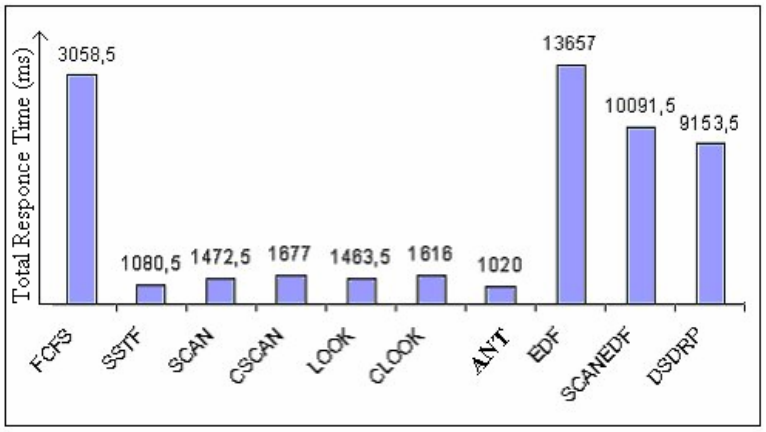


Table 6. Average of algorithm responses for queue size 10 with multiple parameters.

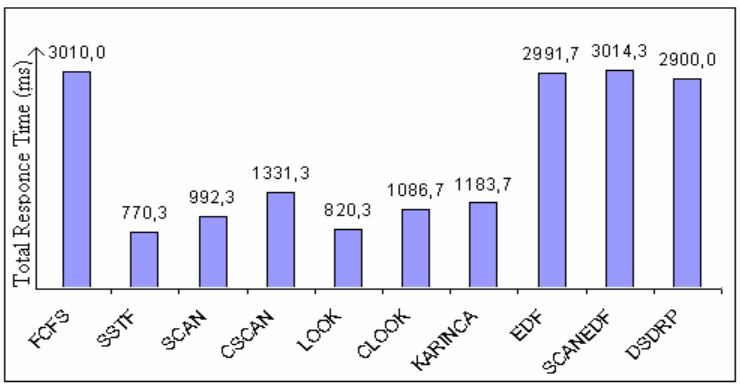


Table 7. Average of algorithm responses for queue size 20 with multiple parameters.

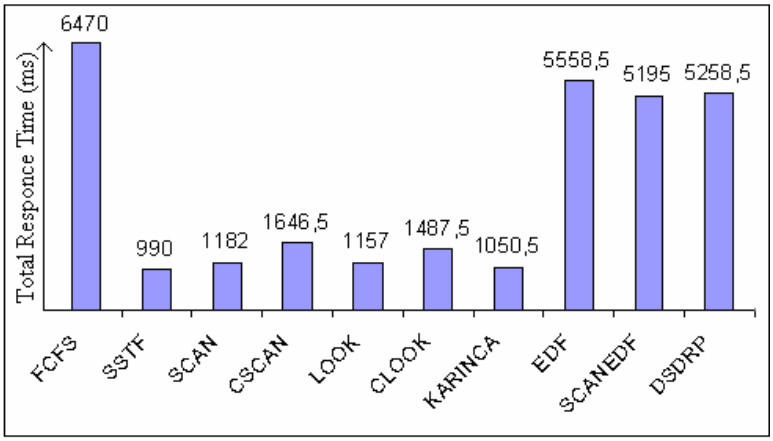
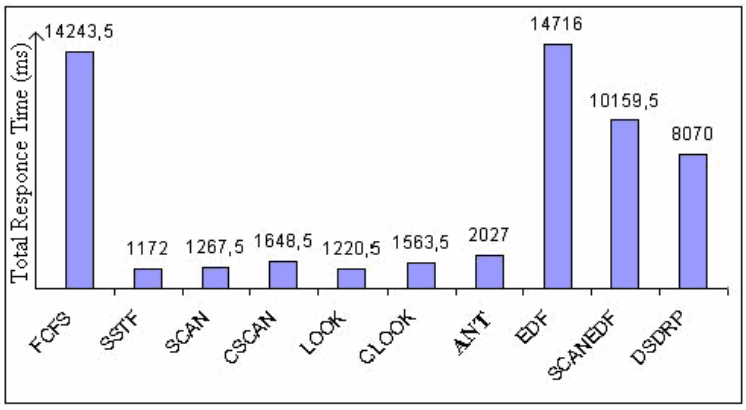


Table 8. Average of algorithm responses for queue size 50 with multiple parameters.



REFERENCES

1. Ruemmler, C. and Wilkes, J., An introduction to disc driving modeling, IEEE Computer, 27(3):17-29 March, 1994.
2. Yeh, T., Kuo, T., Lei, C., and Yen, H., Competitive analysis of on-line disk scheduling. Technical report, Dept. of Electrical Engineering, National Taiwan University, 1995.
3. Andrews, M., Bender, M., and Zhang, L., New algorithms for the disk scheduling problem. In Proceedings of IEEE FOCS, to appear, 1996.
4. Worthington, G. R. Ganger, and Patt, Y. N., Scheduling algorithms for modern disk drives. Proceedings of the ACM Sigmetrics, pages 241--251, May 1994.
5. Aref, W.G., El-Bassyouni, K., Kamel, I., Mokbel, M.F., Scalable QoS-aware disk-scheduling, International Database Engineering and Applications Symposium, Proceedings,17-19 July 2002, pp.256-265.
6. Hartenstein, R., Trends in reconfigurable logic and reconfigurable computing, 9th International Conference on Electronics, Circuits and Systems, Vol. 2, 15-18 Sept. 2002, pp.801–808.
7. Guntsch, M., Middendorf, M., Scheuermann, B., Diessel, O., ElGindy, H., Schmeck, H., So, K., Population based ant colony optimization on FPGA, IEEE International Conference on Field-Programmable Technology, 2002. (FPT). Proceedings. 16-18 Dec. 2002, pp.125-132.

8. Gopalan, K.,and Chiueh, TG., Real-time Disk Scheduling Using Deadline Sensitive SCAN, Technical Report TR-92, Experimental Computer Systems Labs, Dept. of Computer Science, SUNY at Stony Brook, Stony Brook, NY, USA, Jan 2001.
9. Fujitsu Limited : M2361A Mini-Disk Drive Engineering Specifications, Fujitsu Limited, 1984.
10. Dorigo, M., Member, IEEE, Maniezzo, V., and Colorni, A., The Ant System: Optimization by a colony of cooperating agents, IEEE Transactions on Systems, Man, and Cybernetics-Part B, Vol.26, No.1, 1996, pp.1-13.
11. Yongcheng L., See-Mong T., Zhigang C., and Roy H. Campbell: Disk scheduling with dynamic request priorities. Technical report. University of Illinois at Urbana-Champaign, IL, August 1995.
12. Seltzer, M., Chen, P., and Ousterhour, J., Disk scheduling revisited. Winter USENIX Technical Conference (Washington, DC, 22 26 January 1990.