The International Journal of Engineering and Information Technology
journal homepage:www.ijeit.misuratau.edu.ly

Proposing Disk Scheduling Algorithm to Enhance the Efficiency of Disk Performance

Elnaas, G.
ghada_elnaas@cit.edu.ly
The College of Industrial Technology

Sullabi, M.
m.sullabi@it.misuratau.edu.ly
Misurata University

Abstract— **Management of disk scheduling is a very important aspect of operating system. Disk scheduling involves a careful examination of pending requests to determine the most efficient way to serve these requests. A disk scheduler examines the positional relationship among waiting requests, then reorders the queue so that the requests will be serviced with minimum seek. Performance of the disk scheduling completely depends on how efficient is the scheduling algorithm to allocate services to the request in a better manner. After the arrival of the multi-core processor, the speed of the processors became much faster with the Hyper-Threading technology, and with this the speed of the processors increased dramatically with a record speed. All processers need CPU time and read/write time together to complete their execution. Read/write operations require the operating system to access the disk to store and retrieve data. In the recent years many algorithms (FIFO, SSTF, SCAN, C-SCAN, LOOK, etc.) are developed in order to optimize the system disk I/O performance. The purpose of this study is to obtain a new scheduling algorithm that reduces seek time, sum of head movement, spin time, and transfer time, so as to improve disk performance efficiency in a better way to try to synchronize with CPU performance.**

Index Terms— **Disk Scheduling Algorithm, Average Seek Time, Total Head Movement, FCFS, SSTF, SCAN, LOOK, C-SCAN, C-LOOK**

I. INTRODUCTION

In the era of high-performance computing, most of the attention is focused on improving the ability of computers by increasing the speed of their work, as management of disk performance is an important aspect of an operating system. Since the speed of processor and main memory have been increased doubles than the speed of the disk, the difference in speed of processor and disk, Read/write performance of disk has become an importance. In any disk system with a moving read/write head, the seek time between cylinders takes a significant amount of time. This seek time should be minimized to better access time. When generated requests for reading and writing disk records, the operating system handles these read/write requests from the queue and processes them one by one. The algorithm used to choose which read/write request is going to be fulfilled earliest is called disk scheduling algorithm. The main objectives for any disk scheduling algorithm are minimizing the response time and maximizing the throughput. For this reason researchers are trying to improve the performance of traditional scheduling algorithms by trying to reduce the average waiting time and turnaround time. [1]

Received 01 May , 2024; revised 11 May, 2024; accepted 15 Mar 2024.
Available online 08 Aug, 2024.

A. DISK SCHEDULING CRITERIA

- Generally, a set of criteria is established against which various scheduling policies are evaluated, explained as following:
- Seek Time: The time required for the disk arm head to move from one track to another.
- Rotational latency: The time required for the disk to rotate the desired sector to the disk head.
- Transfer time is the time taken to transfer the data from the disk.
- Disk bandwidth: Total number of bytes transferred divided by the total time between the first request for service and the completion of last transfer.[2]

B. DISK SCHEDULING ALGORITHMS

Operating system do disk scheduling to schedule read/write requests arriving to the disk. by Disk Scheduling Algorithms so process can make multiple read/write requests and multiple processes run at the same time. The requests made by a process may be located at different sectors on different tracks. Due to this, seek time may increase more. The Disk Scheduling Algorithms can help in minimizing the seek time by ordering the requests made by a particular arrangement. There are many disk scheduling algorithms such as FCFS, SSTF, SCAN, C-SCAN, and LOOK etc.[1]

- *First Come First Serve (FCFS) Scheduling:*

This is the simplest algorithm, serves the request coming first. It is simple to implement. But it has some disadvantages that it does not provide the fastest service, and the average head movement in the algorithm is too high.

- *Shortest Seek Time First (SSTF) Scheduling:*

Shortest Seek Time Fist (SSTF) selects the request with minimum seek time from the current head position. To compare to FCFS gives substantial improvement.

- *SCAN Scheduling:*

Scan algorithm is called elevator algorithm. In this the disk arm moves from one end of the disk towards other end, it scans reversely servicing the requests that it didn't get earlier. Comparing with FCFS and SSTF it gives better performance.

- *C-SCAN Scheduling:*

C-Scan scheduling algorithm is called *Circular scan.* The head moves from one end to other end of the disk, servicing the request along the way. The waiting time increases in the algorithm.

- *LOOK Scheduling:*

This approach is equivalent to SCAN algorithm, except that the disk arm moves across the full width of the disk. The arm goes as far as the final request in each direction and reverses immediately.

- *C- LOOK Scheduling:*

C-LOOK is an enhanced version of both SCAN as well as LOOK disk scheduling algorithms. But the scanning doesn't go past the last request in the direction that is moving. It too jumps to the other end but not all the way to the end.

II. RELATED WORK DONE

Many research have been done in the topic "Disk Scheduling Algorithms" and many new algorithms are formed on the basis of previous research. This is some of previous research works:

In the recent years many research have been done for enhancing the disk performance we have chosen from them :

Z. Dimitrijevic, R. Rangaswami and E. Y. Chang have presented Semi-perceptible I/O, which divides disk I/O requests into small temporal units of disk commands to improve the perceptibility of disk access. [3]

Cheng - Han Tsai, Tai - Yi Huang, Edward T. - H. Chu, Chun-Hang Wei and Yu - Che Tsai propose a novel real-time disk-scheduling algorithm called WRR - SCAN (Weighted-Round-Robin-SCAN) to provide quality guarantees for all in-service streams encoded at variable bit rates and bounded response times for aperiodic jobs. [4]

Daniel L. Martens and Michael J. Katchabaw developed a new disk scheduling algorithm focuses on dynamic scheduling algorithm selection and tuning.[5]

Mohammod Abul Kashem, Sandipon Saha and Mohammad Naderuzzaman proposed a disk I/O scheduling scheme that can automate the manual configuration and selection of disk schedulers. The scheduling scheme can learn about workloads, file systems, disk systems, CPU systems, and user preferences.[6]

Amar Ranjan Dash, Sandipta Kumar Sahu and B Kewal proposed a new disk scheduling algorithm, MODSBSM. provides better performance metrics by minimizing the average disk access time. The algorithm also able to detect and resolve the bad sectors of hard-disk.[7]

Muhammad Younus Javed, Ihsan Ullah Khan developed a simulator which uses four disk scheduling algorithms (FCFS, SSTF, LOOK for both upward and downward direction, and C-LOOK) to measure their performance in terms of total head movement, 2015.[8]

Sukanya Suranauwarat did a research paper presents an intuitive, engaging, and easy-to-use simulator that animates the concepts of traditional disk scheduling algorithms. The simulator has operating modes: simulation, practice, and comparison, 2017.[9]

Sandipon Saha, Md. Nasim Akhter and Mohammod Abul Kashem presented a new real-time disk scheduler that imposes almost no performance penalty over non-real-time optimal schedulers when given sufficient slack time, 2013.[10]

Avneesh Shankar, Abhijeet Ravat and Abhishek Kumar Pandey identified the benefits and drawbacks of the disk scheduling algorithms and proposing an improved algorithm. The performance of a disk drive depends on various factors like seek time, latency time, access time and structure of the disk. This paper covers the comparative analysis of famous disk scheduling algorithms and proposal of a new algorithm with better performance, 2019.[11]

John Ryan Celis, Dennis Gonzales, Erwinaldgeriko Lagda and Larry Rutaquio Jr discussed the structure of a disk and the hardware activities involved in the retrieval of data on a direct access storage device, 2014. [12]

Akanmu, T. A., Aadegoke, B. O. and Oladoye, S. F presented an Hybridized Disk Scheduling Algorithm (HDSA) which shows better performance than existing conventional disk scheduling algorithms such as FCFS, SSTF, SCAN, C-SCAN, LOOK and C-LOOK), 2019. [13]

Bishwo Prakash Pokharel did a research paper presents a comparative analysis of disk scheduling algorithms , 2021. [14]

III. PROBLEM DOMAIN

Recent improvements in hard disk technology have increased storage capacities of hard disk drives by 60% to 80% annually, but disk performance improvements have been lacking increasing at only 7% to 10 % annually. After the arrival of the multi-core processor, the speed of the processors became much faster with the Hyper-Threading technology, and with this the speed of the processors increased dramatically with a record speed of 40% to 60% doubling annually. All processers need CPU time and read/write time together to complete their execution. Read/write operations require the operating system to access the disk to store and retrieve data. That's why the discrepancy between CPU and hard disk performance must continue to be addressed.[15][16]

IV. PROPOSED DISK SCHEDULING ALGORITHM

The seek time in operating systems is very important. That's because all device requests are linked in queues, For this reason, increasing the length of the seek time slows down the system as a whole. In the proposed algorithm, we tried to reduce the seek time by setting a specific format for the tracks to keeping Head Movements (tracks) to the least amount as possible. So the main aim of our proposed disk scheduling algorithm is to improve the disk performance through reducing average seek time of the disk scheduling algorithm. So there will be a faster data transfer. The main goal behind all is to enhance the system performance.

A. THE PROPOSED ALGORITHM APPROACH

The proposed algorithm approach goes several steps as shown in the figure(1) as following :

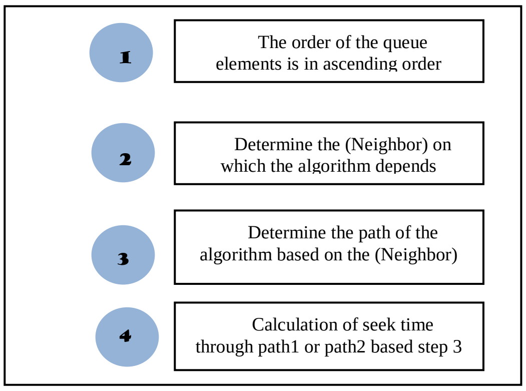
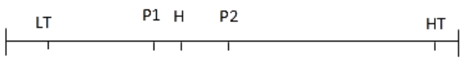

1
The order of the queue elements is in ascending order
2
Determine the (Neighbor) on which the algorithm depends
3
Determine the path of the algorithm based on the (Neighbor)
4
Calculation of seek time through path1 or path2 based step 3

Figure 1. Steps for proposed algorithm approach.

1) *Queue items order:* When the list of pending tasks arrives, its elements are arranged according to their arrival. The first thing dose is letting the request matrix represent a matrix that stores the indexes of the tracks that have been requested, along with the header track, which is the position of the disk head, then arrange them in ascending order to be dealt with while they are in order.
2) *Determine the (Neighbor):* The proposed algorithm is based on the adjacent track of the disk head closest to one of the two ends of the disk and is calculated as shown:



A =(P1-LT)
B=(HT-P2)
if A<B
N = P1 or N= P2

whereas:
LT : Lowest Track
HT : Height Track
H : Head Position
P1 :Track before the head position
P2 : Track after the head position
N : Neighbor track head
A : Distance between P1 and LT
B : Distance between P2 and HT

3) *Determine the path of the algorithm:* After specifying the path adjacent to the header that meets the aforementioned conditions in the previous point, now define the path of the algorithm based on it. If the neighbor is the track(P1), we follow the following steps:

   1- Find the positive distance from vertex to (P1). Summation with it the distance from the track(P1) to the lowest track and put it in the seek counter.
   2- Find a track from the required array that has not been accessed.
   3- Increase the total distance to the seek counter.
   4- Move to step number 2 until all tracks are calculated.

   If the neighbour is the track (P2), follow these steps:

   1- Find the positive distance from vertex to (P2). Summation with it the distance from the track(P2) to the height track and put it in the seek counter.
   2- Find a track from the required array that has not been accessed.
   3- Increase the total distance to the seek counter.
   4- Move to step number 2 until all tracks are calculated.
4) *Calculation of seek time:* finally will be calculated seek time through selected path in precedent step. Figure (2) shows the paths of the algorithm.

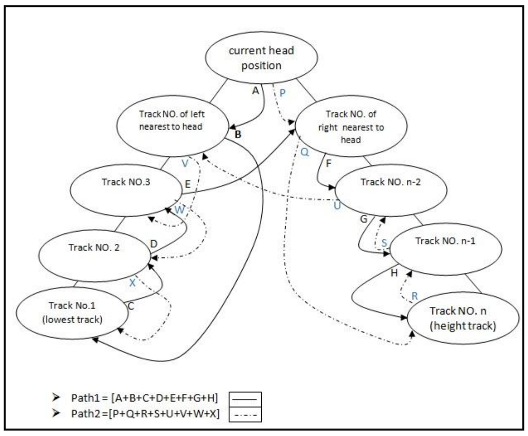

current head position
Track NO. of left nearest to head
Track NO. of right nearest to head
Track NO.3
Track NO. 2
Track No.1 (lowest track)
Track NO. n-2
Track NO. n-1
Track NO. n (height track)
A
B
C
D
E
F
G
H
P
Q
R
S
U
V
W
X
Path1 = [A+B+C+D+E+F+G+H]
Path2 =[P+Q+R+S+U+V+W+X]

Figure2. The paths of proposed algorithm.

B. *EXPERIMENTS PERFORMED*

To evaluate the performance of our proposed algorithm, taken three different samples. In each case, the experimental results of proposed algorithm for each sample is compared with other disk scheduling algorithms.

sample I: Suppose the head of a moving – head disk with 200 tracks numbered 0 to 199), current position of I/O head is : 53 and the order of request in queue requests is(89,183,37,122,14,124,65,67).[1] Table 1 shows the comparison of all the algorithms with proposed algorithm. Figure 3, Figure 4, Figure 5, Figure 6, Figure 7, Figure 8 and Figure 9 show the representation of Proposed Algorithm, FCFS, SSTF, SCAN, C-SCAN, LOOK and C-LOOK respectively. Figure 10 and Figure 11 show the comparison of total head movements and average seek time respectively.

TABLE I. Comparison of all algorithms with proposed algorithm(sample I).

Algorithm	THM	AST
Proposed Algorithm	177	22.125ms
FCFS	640	80ms
SSTF	236	29.5ms
SCAN	236	29.5ms
C-SCAN	382	47.75ms
LOOK	299	37.37ms
C-LOOK	322	40.25ms

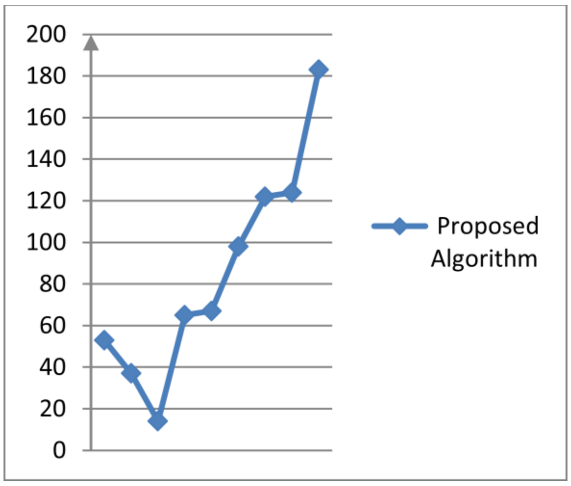


Figure 3. Representation of proposed algorithm (Sample I).

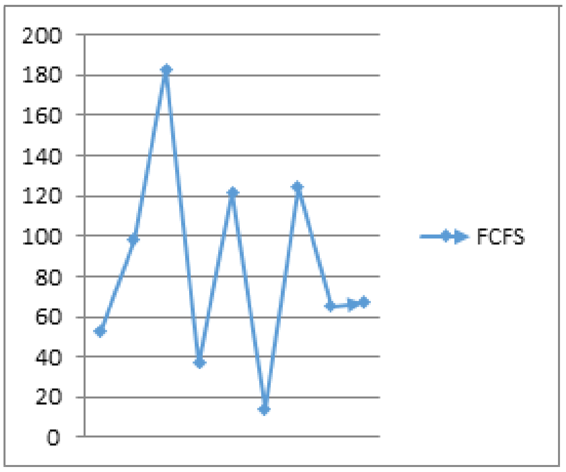


Figure 4. Representation of FCFS (Sample I).

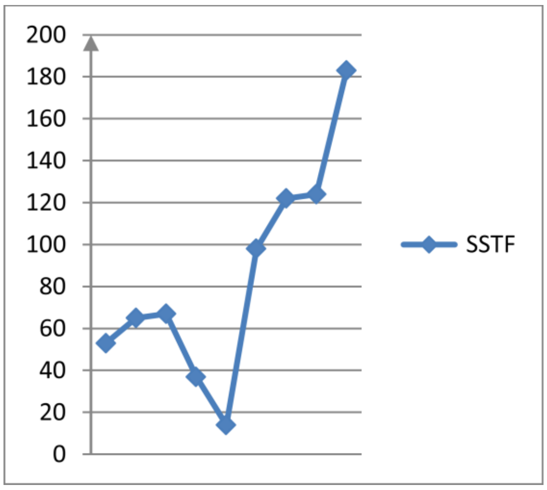


Figure 5. Representation of SSTF (Sample I).

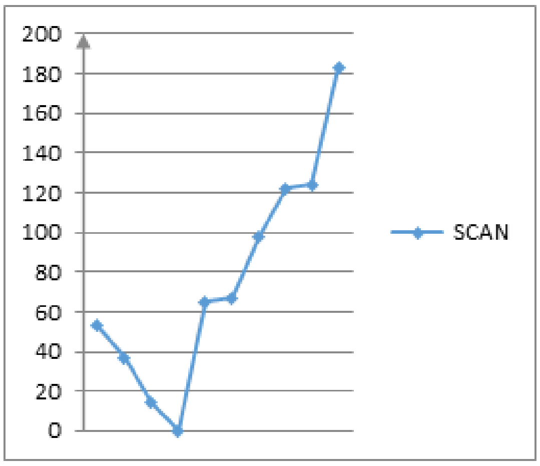


Figure 6. Representation of SCAN (Sample I).

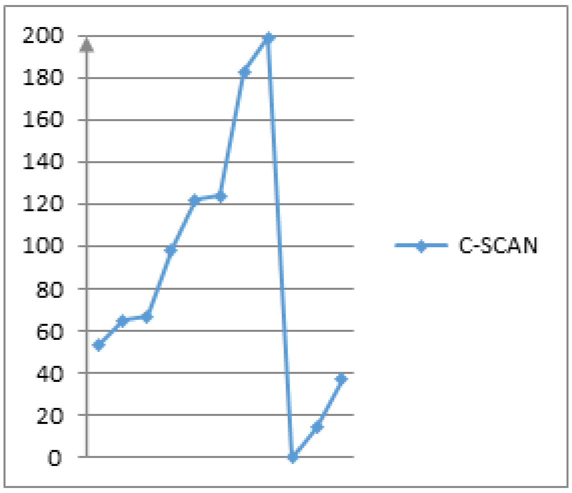


Figure 7. Representation of C-SCAN (Sample I).

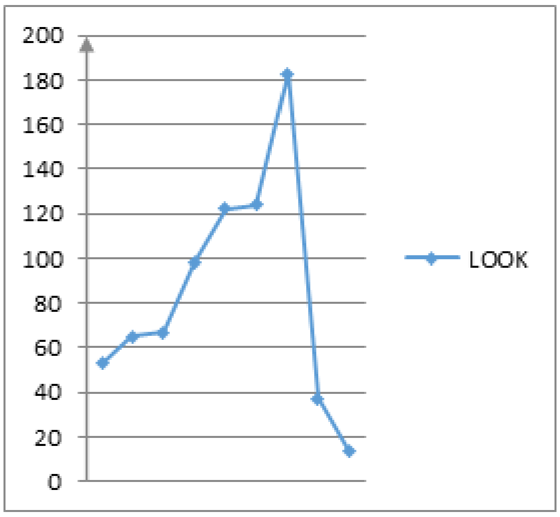


Figure 8. Representation of LOOK (Sample I).

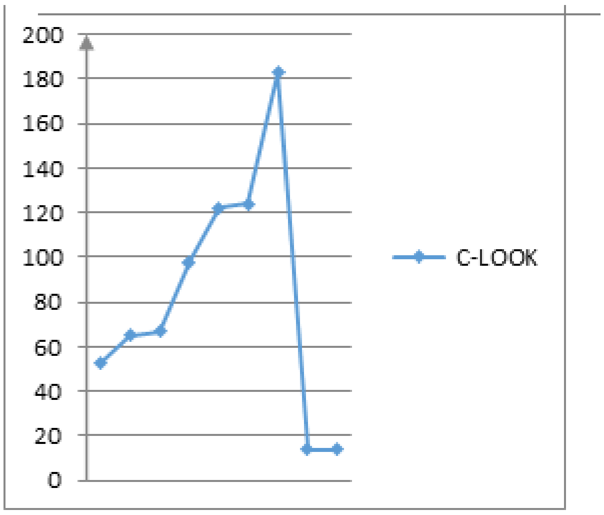


Figure 9. Representation of C-LOOK (Sample I).

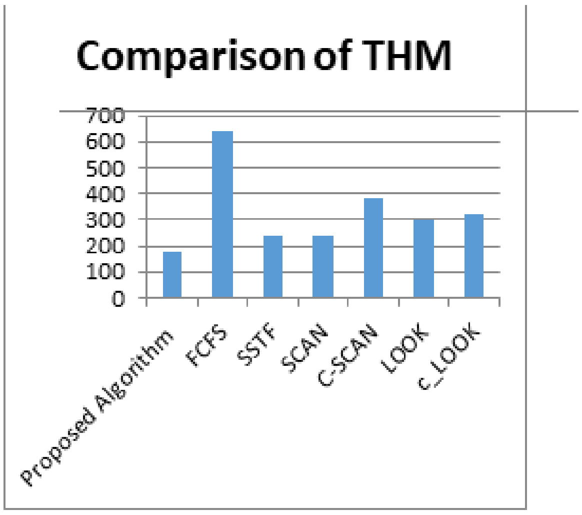


Figure 10. Comparison of Total Head Movement (Sample I).

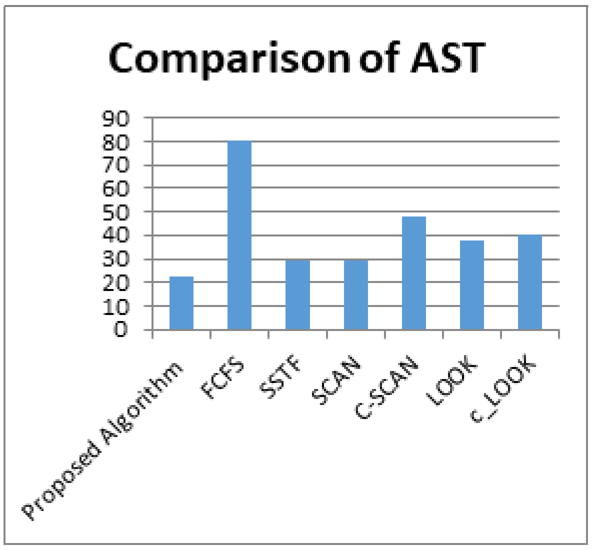


Figure 11.Comparison of Average Seek Time (Sample I).

sample II : Suppose the head of a moving – head disk with 201tracks numbered 0 to 200), current position of Read/Write head is : 59 and the order of request in queue requests (100,180,40,120,20,130,60,70)[15].Table 2 shows the comparison of all the algorithms with proposed algorithm. Figure 12, Figure 13, Figure 14, Figure 15, Figure 16, Figure 17 and Figure 18 show the representation of Proposed Algorithm, FCFS, SSTF, SCAN, C-SCAN, LOOK and C-LOOK respectively. Figure 19 and Figure 20 show the comparison of total head movements and average seek time respectively.

TABLE 2. Comparison of all algorithms with proposed algorithm(sample II).

Algorithm	**THM**	**AST**
Proposed Algorithm	199	24.87ms
FCFS	631	78.87ms
SSTF	230	28.75ms
SCAN	321	40.125ms
C-SCAN	381	47.625ms
LOOK	281	35.125ms
C-LOOK	301	37.625ms

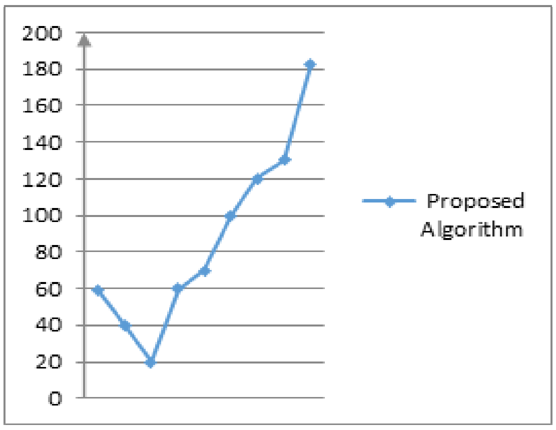


Figure 12. Representation of proposed algorithm (Sample II).

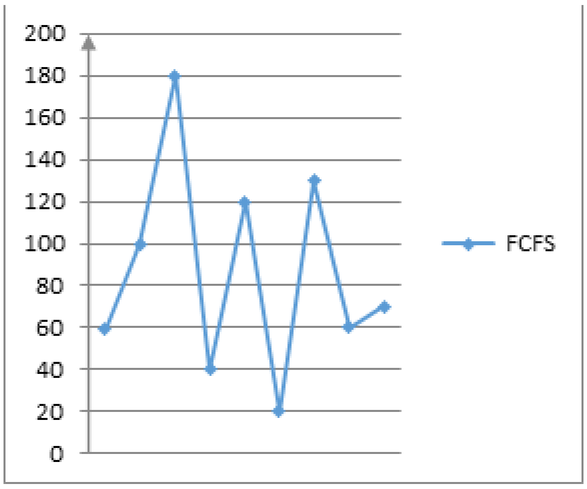


Figure 13. Representation of FCFS (Sample II).

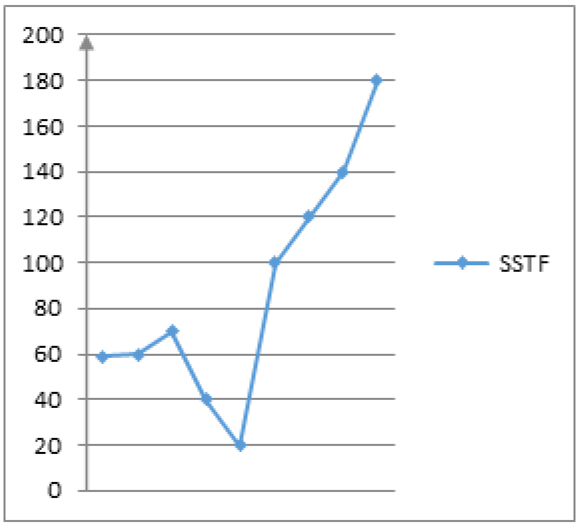


Figure 14. Representation of SSTF (Sample II).

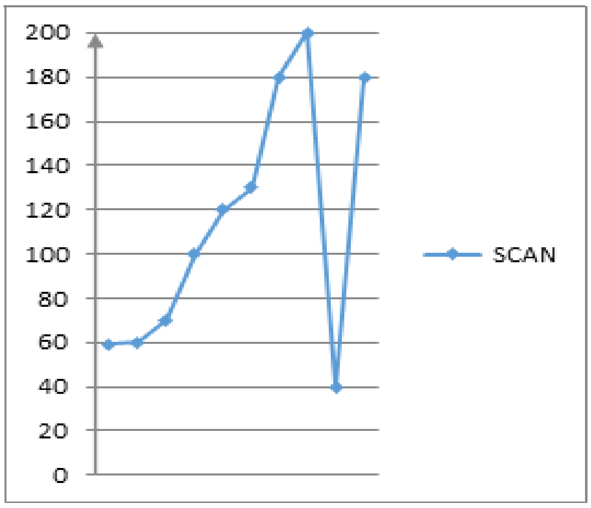


Figure 15. Representation of SCAN (Sample II).

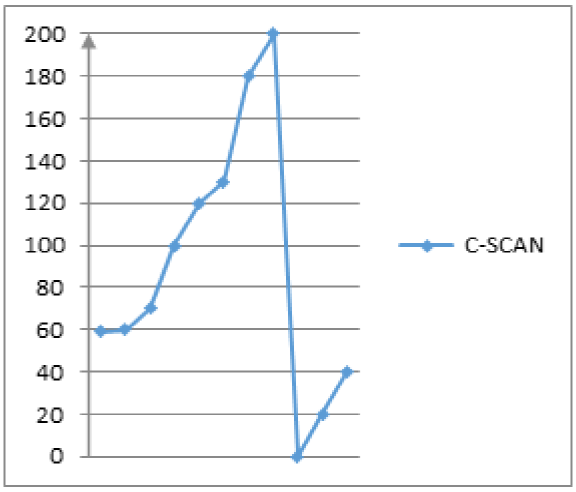


Figure 16. Representation of C-SCAN (Sample II).

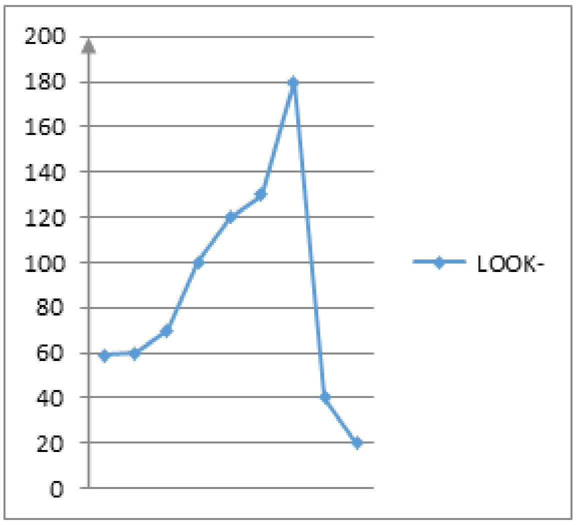


Figure 17. Representation of LOOK (Sample II).

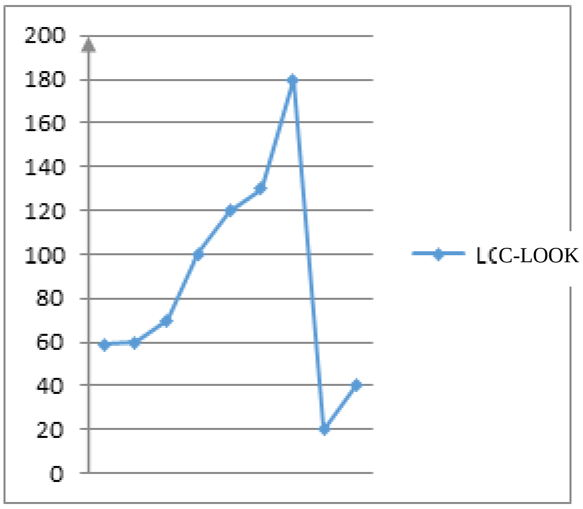


Figure 18. Representation of C-LOOK (Sample II).

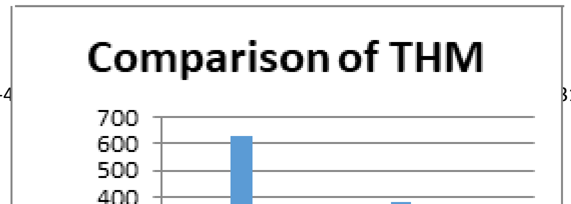


Figure 19. Comparison of Total Head Movement (Sample II).

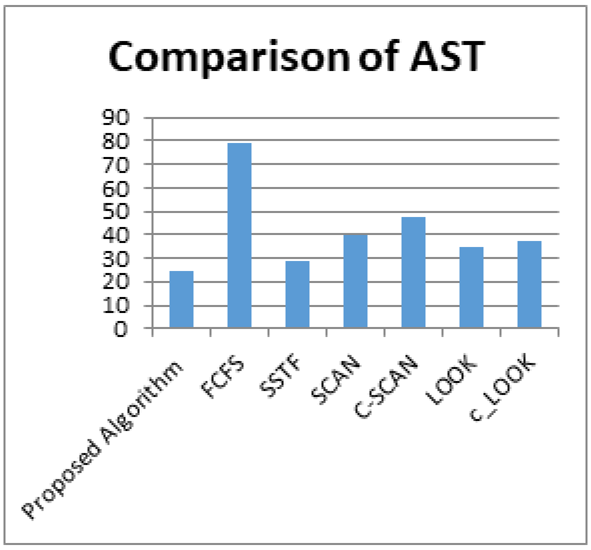


Figure 20. Comparison of Average Seek Time (Sample II).

sample III : Suppose the head of a moving – head disk with 200 tracks numbered 0 to 1024), current position of I/O head is : 354 and the order of request in queue requests (354,703,477,882,692,510,783,987,648,641,390,120,656,72,320,601,236,580,302,383).[11] Table 3 shows the comparison of all the algorithms with proposed algorithm. Figure 21, Figure 22, Figure 23, Figure 24, Figure 25, Figure 26 and Figure 27 show the representation of Proposed Algorithm, FCFS, SSTF, SCAN, C-SCAN, LOOK and C-LOOK respectively. Figure 28 and Figure 29 show the comparison of total head movements and average seek time respectively.

TABLE 3. Comparison of all algorithms with proposed algorithm(sample III).

Algorithm	THM	AST
Proposed Algorithm	1197	63ms
FCFS	5403	284.36ms
SSTF	1269	66.78ms
SCAN	1341	70.57ms
C-SCAN	1929	101.52ms
LOOK	1537	80.89ms

C-LOOK	1802	94.78ms

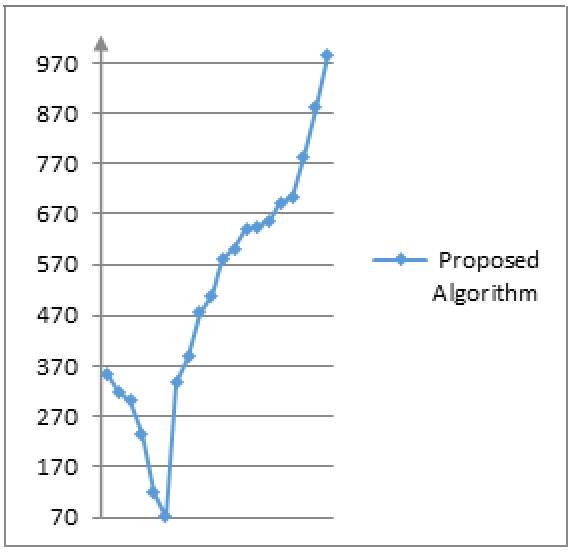


Figure 21. Representation of proposed algorithm (Sample III).

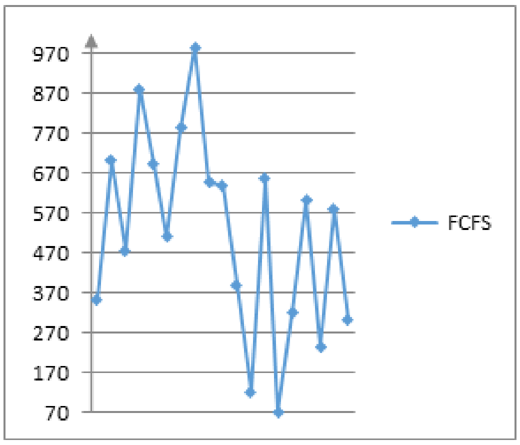


Figure 22. Representation of FCFS (Sample III).

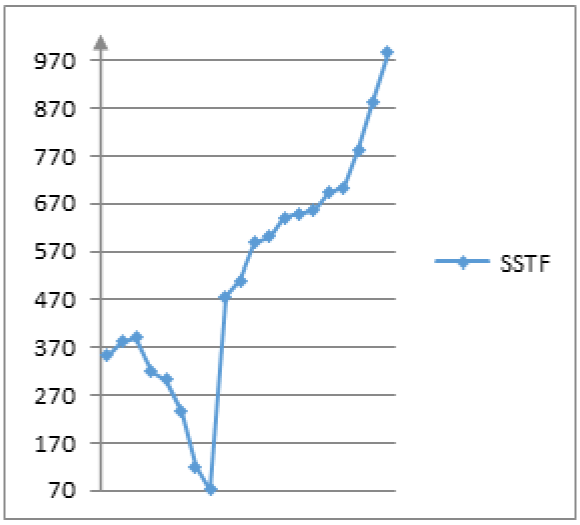


Figure 23. Representation of SSTF (Sample III).

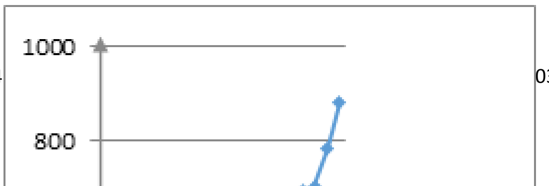


Figure 24. Representation of SCAN (Sample III).

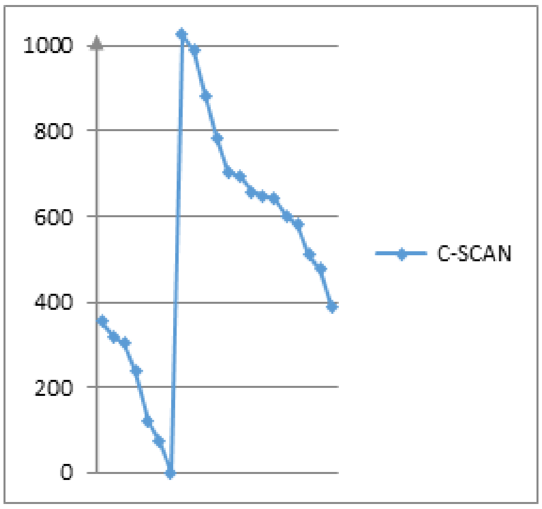


Figure 25. Representation of C-SCAN (Sample III).

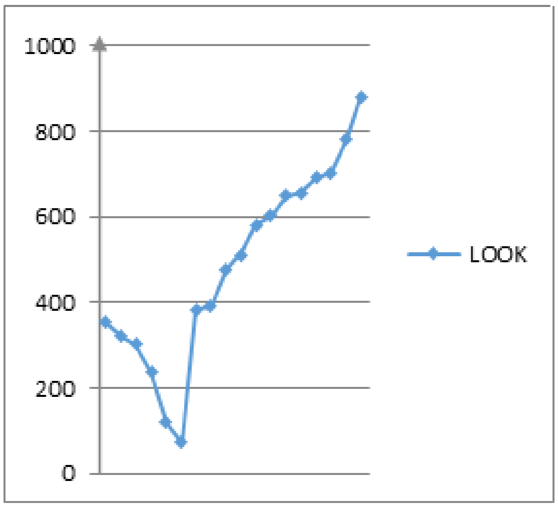


Figure 26. Representation of LOOK (Sample III).

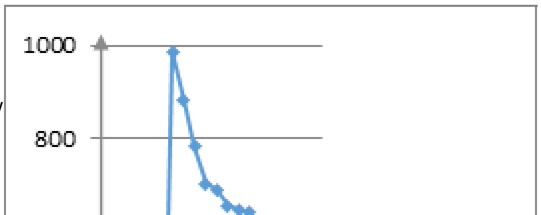


Figure 27. Representation of C-LOOK (Sample III).

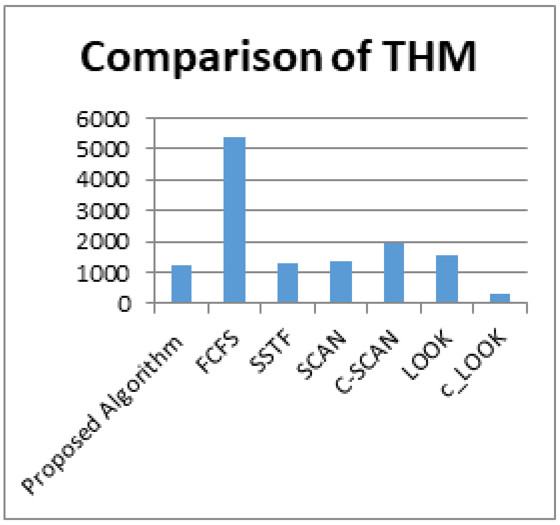


Figure 28. Comparison of Total Head Movement (Sample III).

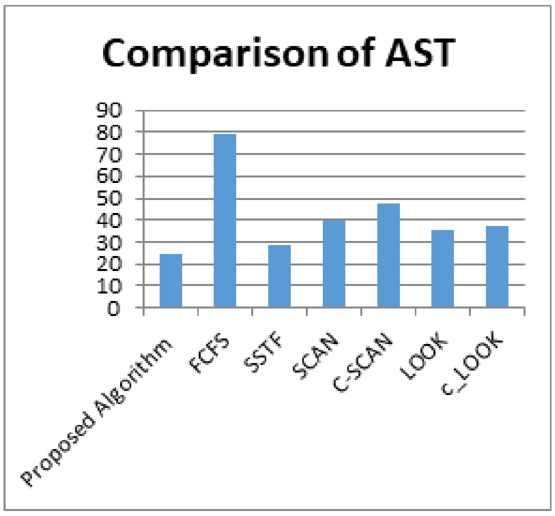


Figure 29. Comparison of Average Seek Time (Sample III).

CONCLUSION

In conclusion, we have presented proposed disk scheduling algorithm was showed better performance than other disk scheduling algorithms (FIFO, SSTF, SCAN, C-SCAN and LOOK). From the above experiment and comparison of proposed algorithm with traditional algorithms explained that the proposed algorithm average seek time and head movement has been reduced which Increased the efficiency of the disk performance.

REFERENCES

[1] Abraham Silberschatz, Peter B. Galvin, and Gerg Gagne, (2014) ―Operating System Concepts‖. Willey India 9th Edition .

[2]Chris Ruemmler and John Wilkes, (1994)"An Introduction to Disk Drive Modling", IEEE Transactions on Computers.

[3] Z. Dimitrijevic, R. Rangaswami and E. Y. Chang, “Support for Preemptive Disk Scheduling”, IEEE Transactions on computers, Vol. 54, No. 10, Oct 2005.

[4] C. Tsai, T. Huang, E. Chu, C. Wei and Y. Tsai, “An Efficient Real-Time Disk-Scheduling Framework with Adaptive Quality Guarantee”, IEEE Transactions on computers, Vol. 57, No. 5, May 2008.

[5] Daniel L. Martens and Michael J. Katchabaw," Optimizing System Performance Through Dynamic Disk Scheduling Algorithm Selection", Wseas Trans. Information Science and Application,2006

[6] Mohammod Abul Kashem, Sandipon Saha, Mohammad Naderuzzaman,(2013),"A New Approach of Disk Scheduling Algorithm", NCICIT 2013: 1st National Conference on Intelligent Computing and Information Technology,November 21, CUET, Chittagong-4349, Bangladesh.

[7] Amar Ranjan Dash, Sandipta Kumar Sahu, B Kewal,(2019)," An Optimized Disk Scheduling Algorithm With Bad-Sector Management", International Journal of Computer Science, Engineering and Applications (IJCSEA) Vol. 9, No.1/2/3, June 2019.

[8] Muhammad Younus Jave and Ihsan Ullah Khan,(2015)," Simulation and performance comparison of four duling disk schealgorithms ", IEEE Transactions on computers, April 2015.

[9] Sukanya Suranauwarat, (2017)," A Disk Scheduling Algorithm Simulator ", Computers In Education Journal, Vol.8, Issue 3, September 2017.

[10] Sandipon Saha, Md. Nasim Akhter and Mohammod Abul Kashem,(2013)," A New Heuristic Disk Scheduling Algorithm", International Journal Of Scientific and Technology Research Vol. 2, Issue 1, January 2013

[11] Avneesh Shankar, Abhijeet Ravat and Abhishek Kumar Pandey,(2019)," Comparative Study of Disk Scheduling Algorithms and Proposal of a New Algorithm for Better Efficiency", 2nd International Conference On Advanced Computing And Software Engineering (ICACSE-2019)

[12] John Ryan Celis1, Dennis Gonzales2, Erwinaldgeriko Lagda3 and Larry Rutaquio Jr.4,(2014)," A Comprehensive Review for Disk Scheduling Algorithms", IJCSI International Journal of Computer Science Issues, Vol. 11, Issue 1, No 1, January 2014

[13] Akanmu, T. A., Aadegoke, B. O. and Oladoye, S. F,(2019)," An Hybridized Disk Scheduling Algorithm (HDSA)", International Journal of Scientific and Research Publications, Vol. 9, Issue 3, March 2019.

[14] Bishwo Prakash Pokharel,(2021)," A Comparative Analysis of Disk Scheduling Algorithms", International Journal for Research in Applied Sciences and Biotechnology Vo.8, Issue-2, March 2021.

[15] C.Mallikarjuna and P.Chitti Babu, (2016), " Performance Analysis of Disk Scheduling Algorithms", International Journal of Computer Sciences and Engineering Vo.4, Issue-5, May2016.

[16] Ahalya R, Chethana S, Varsha A ,” A Comprehensive Analysis of Disk Scheduling Algorithms”, International Journal of Advances in Engineering and Management (IJAEM) Vo 2, Issue 9,Novamber 2020.