Journal of Scientific and Human Dimensions
مجلة الأبعاد العلمية والإنسانية
Volume 1, Issue 2, 2025, Pages: 1238-1246
Journal homepage: https://jshd.com.ly/index.php/jshd/en

Performance Analysis and Evaluation of Disk Scheduling Algorithms in Operating Systems

Somaya Salam Omar [1*] , Ali Salam Alatrsh[2], Noura M. Alkurri[3]
[1,2] Department Electric and Electronic, College of Engineering, Nalut University, jado, Libya
[3] Computer Science Department, Faculty of Science Assaba, University of Ghryan, Assaba, Libya

تحليل وتقييم أداء خوارزميات جدولة الأقراص الصلبة في نظم التشغيل

سمية سالم عمر[1*] ، علي سالم الاطرش[2*]، نورا محمد الكري[3*]
[1,2] قسم الهندسة الكهربائية والإلكترونية، كلية الهندسة جادو، جامعة نالوت، ليبيا
[3] قسم الحاسب الألى، جامعة غريان، كلية العلوم الاصابعة، الاصابعة، ليبيا

*Corresponding author: s.omar@nu.edu.ly

Received: April 05, 2026	Accepted: May 25, 2026	Published: June 30, 2026

Abstract:
Optimal management of hardware resources is a critical cornerstone of modern operating system design. The efficiency of secondary storage systems, particularly Hard Disk Drives (HDDs), directly influences overall system performance, heavily relying on key metrics such as access time and data transfer bandwidth. In contemporary multiprogramming environments, multiple Input/Output (I/O) requests simultaneously pool in the system memory. When the disk drive is occupied, a specialized scheduling mechanism is required to structure and manage the request queue effectively. This paper presents a comprehensive analytical and comparative study on the dynamic performance of six prominent disk scheduling algorithms: First-Come, First-Served (FCFS), Shortest Seek Time First (SSTF), SCAN, Circular SCAN (C-SCAN), LOOK, and Circular LOOK (C-LOOK). The methodology relies on evaluating performance efficiency by calculating the total head movement (measured in tracks). The quantitative results reveal significant variations in seek optimization and response times, providing developers and system architects with distinct benchmarks to select the optimal algorithm based on system workloads and operational requirements.

Keywords: Disk Scheduling, Operating Systems, Seek Time, Request Queue, Elevator Algorithm, Hardware Optimization.

الملخص
تُعد الكفاءة الاستثنائية في إدارة الموارد المادية (Hardware) من الركائز الأساسية التي تقوم عليها نظم التشغيل الحديثة . ويتأثر أداء وحدات التخزين الثانوية (كالأقراص الصلبة) بشكل مباشر بزمن الوصول (Access Time) ونطاق حزمة البيانات المرسلة (Bandwidth). نظرًا لاعتماد النظم الحالية على تقنيات البرمجة المتعددة (Multiprogramming) ، تتراكم طلبات القراءة والكتابة (I/O Requests) داخل ذاكرة النظام في آنٍ واحد، مما يتطلب آلية جدولة ذكية لتنظيم صفوف الانتظار (Request Queue) عندما يكون المشغل مشغولاً .تستعرض هذه الورقة بالتفصيل والتحليل والمقارنة الأداء الديناميكي لستة من أبرز خوارزميات جدولة الأقراص وهيFCFS :، SSTF، SCAN، C-SCAN، LOOK، و C-LOOK. وترتكز المنهجية على قياس الفعالية من خلال حساب المسافة الإجمالية لحركة رأس القراءة والكتابة (Total Head Movement)، حيث تُظهر النتائج الكمية تباينًا ملحوظًا في الأداء وسرعة الاستجابة، مما يمنح المطورين رؤية واضحة لاختيار الخوارزمية الأنسب وفقًا لطبيعة الاستخدام ونوع النظام .

الكلمات المفتاحية: جدولة الأقراص، نظم التشغيل، زمن البحث (Seek Time)، صف الطلبات، خوارزمية المصعد.

1. Introduction

Disk schedulingM often referred to as Input / Output (I/O) scheduling—is the process of managing and prioritizing the order in which read and write requests to secondary storage devices are executed. Hard disk drives, due to their mechanical nature, remain one of the slowest hardware components in computer architecture compared to high-speed CPUs and volatile memory (RAM). Consequently, inefficient disk access easily becomes a performance bottleneck, degrading the throughput of the entire operating system(Silberschatz et al., 2018).

The critical demand for efficient disk scheduling is dictated by two main physical and architectural constraints:

- **Concurrence of Processes:** Modern operating systems handle a continuous influx of overlapping I/O requests generated by multiple running processes. Since the disk controller can only service a single request at any given microsecond, pending requests must be ordered systematically within a waiting queue (Geist & Daniel, 1987).
- **Mechanical Track Dispersion:** Requested data sectors are often scattered across distant physical tracks on the disk platters. Arbitrary or unscheduled access patterns cause excessive back-and-forth oscillation of the disk arm, which exponentially increases mechanical wear and prolongs seek latencies(Jacobson & Wilkes, 1991).

To quantitatively evaluate disk performance, three primary temporal parameters must be accounted for:

1. **Seek Time:** The time required to physically move and position the disk arm's read/write head onto the target track.
2. **Rotational Latency:** The duration required for the disk platter to rotate until the requested sector aligns directly beneath the read/write head.
3. **Transfer Time:** The actual time window during which data is transferred from the disk medium to system memory, or vice versa(Paida, 2013).

The cumulative **Total Access Time** is formally expressed by the following equation:

$$\text{Total Access Time} = \text{Seek Time} + \text{Rotational Latency} + \text{Data Transfer Time}$$

Since rotational latency and transfer time are largely governed by the mechanical RPM (Rotations Per Minute) and hardware interface limits, disk scheduling algorithms focus primarily on minimizing the Seek Time by optimizing the total distance traversed by the disk head(Yashvir & Prakash, 2012).

---

2. Methodology and Case Study

To conduct a rigorous comparative analysis, a standardized empirical scenario was established. We assume a disk structure containing 200 tracks, indexed from 0 to 199.

- **Input Request Queue:** $[93, 176, 42, 148, 27, 14, 180]$
- **Initial Read/Write Head Position:** Track 55.

The six selected scheduling algorithms were independently mapped to this dataset to assess their operational efficacy based on the **Total Head Movement (THM)** metric:

2.1 First-Come, First-Served (FCFS)

FCFS is the simplest disk scheduling algorithm. As the name suggests, the requests are addressed in the order they arrive in the disk queue(GeeksforGeeks, n.d.).Using this algorithm doesn’t provide the best results [3].
The FCFS algorithm executes requests sequentially, strictly adhering to their chronological arrival order in the queue without any prior reordering.

- **Advantages:** Exceptionally simple to implement, consumes negligible computational overhead, and ensures absolute fairness, thereby preventing request starvation(Javed & Khan, 2000).
- **Disadvantages:** Prone to severe arm-swinging overhead, resulting in massive head movements and poor seek time efficiency (Yashvir & Prakash, 2012).

- **Mathematical Calculation:**

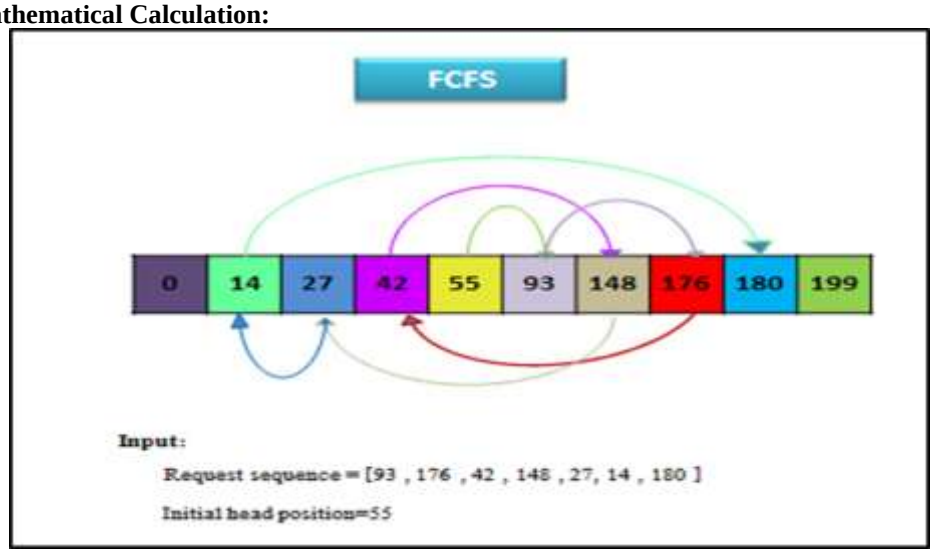

Figure 1: shows inputs of the FCFS algorithm

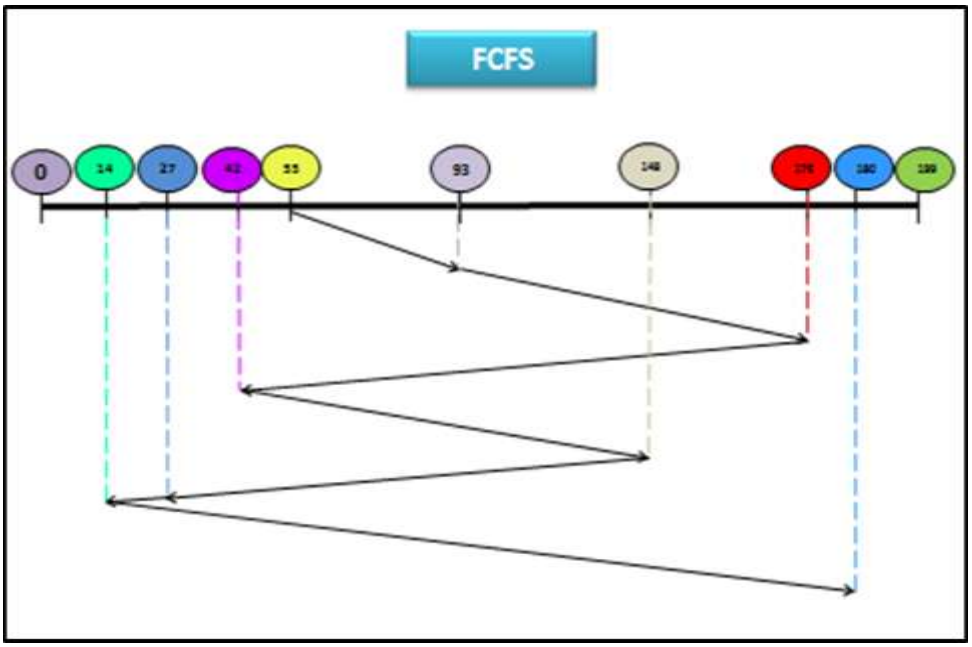

Figure 2: The chart shows the sequence in which requested tracks are serviced using FCFS. The total seek count is calculated as:
(93-55)+(176-93)+(176-42)+(148-42)+(148-27)+(27-14)+(180-14)=661

2.2 Shortest Seek Time First (SSTF)

In SSTF, requests having the shortest seek time are executed first. So, the seek time of every request is calculated in advance in the queue and then they are scheduled according to their seek time(GeeksforGeeks, n.d.). As a result, request near the disk head position will be executed first i. This algorithm gives a substantial improvement in performance compared to FCF, but is not an optimal one (Kumar & Rajendra, 2015). There is a great chance that starvation would take place.[8] The reason for this is if there were a lot of requests close to each other the other requests will never be handled since the distance will always be greater..

SSTF dynamically evaluates the request queue and selects the pending request located closest to the current position of the read/write head, minimizing immediate seek distance.

- **Advantages:** Significantly increases overall system throughput and reduces the average response time for clustered requests(Javed & Khan, 2000).
- **Disadvantages:** Introduces the risk of indefinite starvation for requests located on distant tracks if the system continuously experiences a high volume of localized, nearby I/O requests(Geist & Daniel, 1987).

- **Mathematical Calculation:**

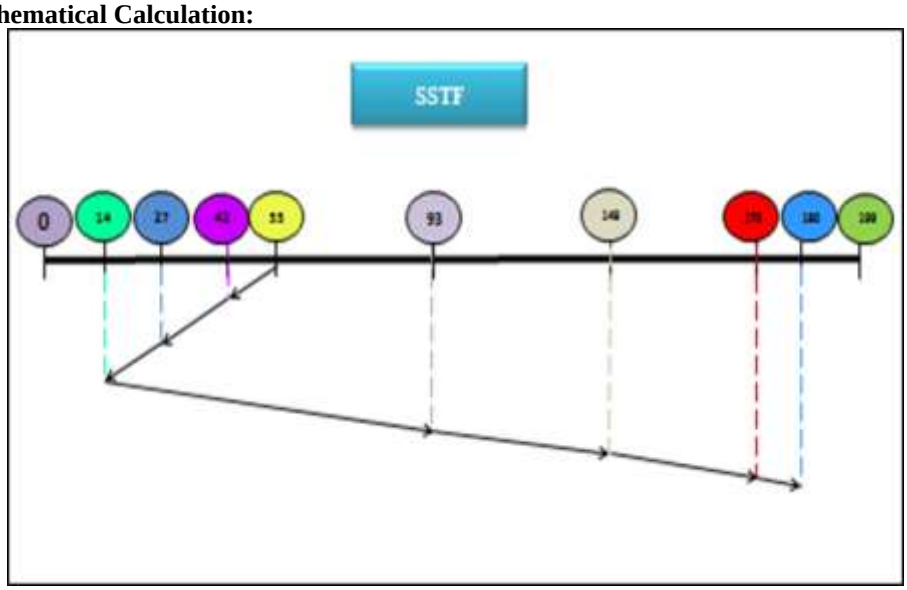

Figure 3: shows inputs of the SSTF algorithm

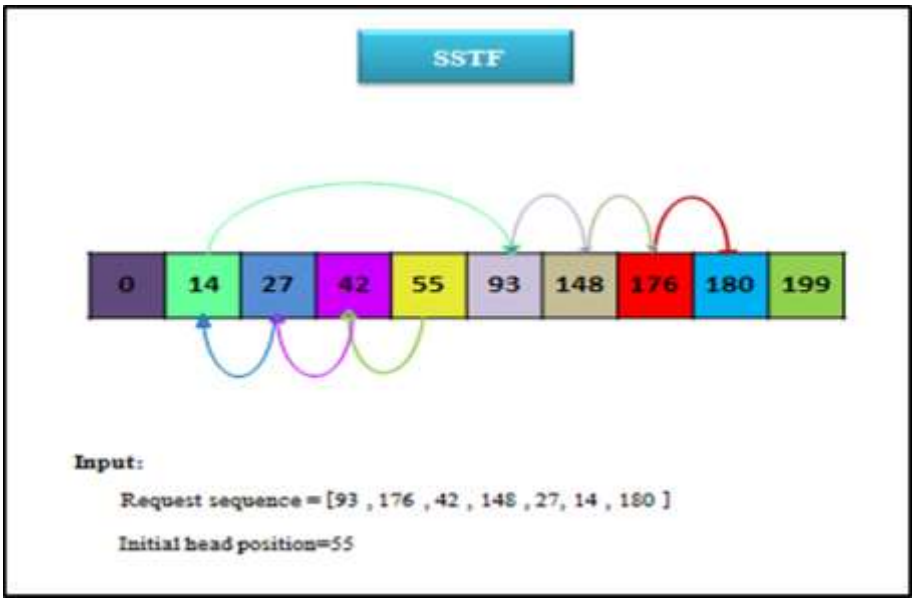

Figure 4: The chart shows the sequence in which requested tracks are serviced using SSTF..The total seek count is calculated as:

(55-42)+(42-27)+(27-14)+(93-14)+(148-93)+(176-148)+(180-176)=207

2.3 SCAN (The Elevator Algorithm)

This algorithm scans the disk back and forth. The disk Head starts from one end of the disk and move towards the other end servicing all the requests in between. After reaching the other end, the head reverses its direction and move towards the starting end servicing all the requests in between(Silberschatz et al., 2018).This SCAN algorithm is also called the elevator algorithm because the behavior of the disk arm is the same as an elevator in a building (Paida, 2013).

Under the SCAN policy, the disk arm moves continuously in a specific direction (e.g., toward the highest track), servicing all requests along its path. Upon reaching the physical boundary of the disk, it reverses its direction and services requests in the opposite direction.

- **Advantages:** Yields high throughput and completely eliminates starvation (Javed & Khan, 2000).
- **Disadvantages:** Forces the mechanical arm to travel all the way to the absolute end of the disk (track 199 or 0), even if no pending requests exist at those boundary tracks (Yashvir & Prakash, 2012).
- **Mathematical Calculation (assuming initial direction toward higher tracks):**

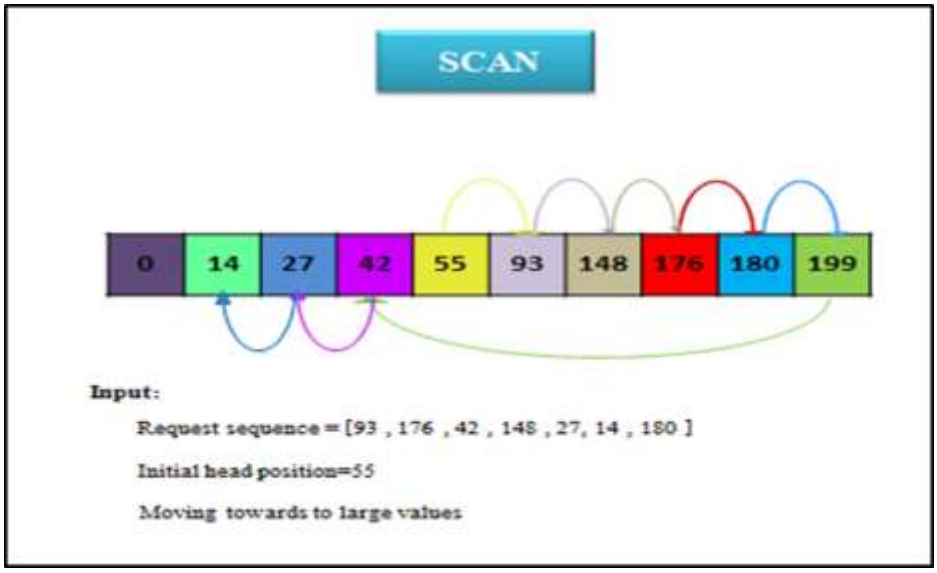

Figure 5: shows inputs of the SCAN algorithm

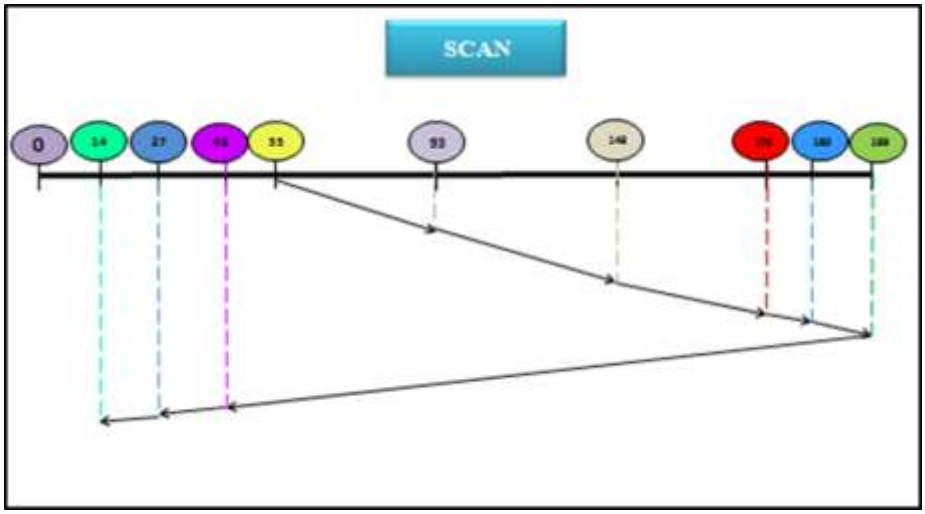

Figure 6: The chart shows the sequence in which requested tracks are serviced using SCAN. The total seek count is calculated as:
(93-55)+(148-93)+(176-148)+(180-176)+(199-180)+(199-42)+(42-27)+(27-14)
=329

2.4 Circular SCAN (C-SCAN)

Circular SCAN (C-SCAN) scheduling algorithm is a modified version of SCAN disk scheduling algorithm. Like SCAN (Elevator Algorithm) C-SCAN is scan- ning in one direction servicing all the requests to the other end. As soon as the head reaches the other end, it immediately returns to the beginning of the disk without servicing any requests on the return trip and starts servicing again once it reaches the beginning(Silberschatz et al., 2018).This is also known as the "Circular Elevator Algorithm" (Paida, 2013).

C-SCAN is a variant designed to provide a more uniform waiting time. The head moves in one direction servicing requests until it reaches the disk boundary; it then immediately snaps back to the opposite beginning boundary (track 0) without servicing any requests during the return trip.

- **Advantages:** Provides highly uniform and predictable waiting times across all disk sectors(Javed & Khan, 2000).
- **Disadvantages:** Increases the overall physical head movement due to the unserviced, complete return sweep to the starting edge(Yashvir & Prakash, 2012).

- **Mathematical Calculation:**

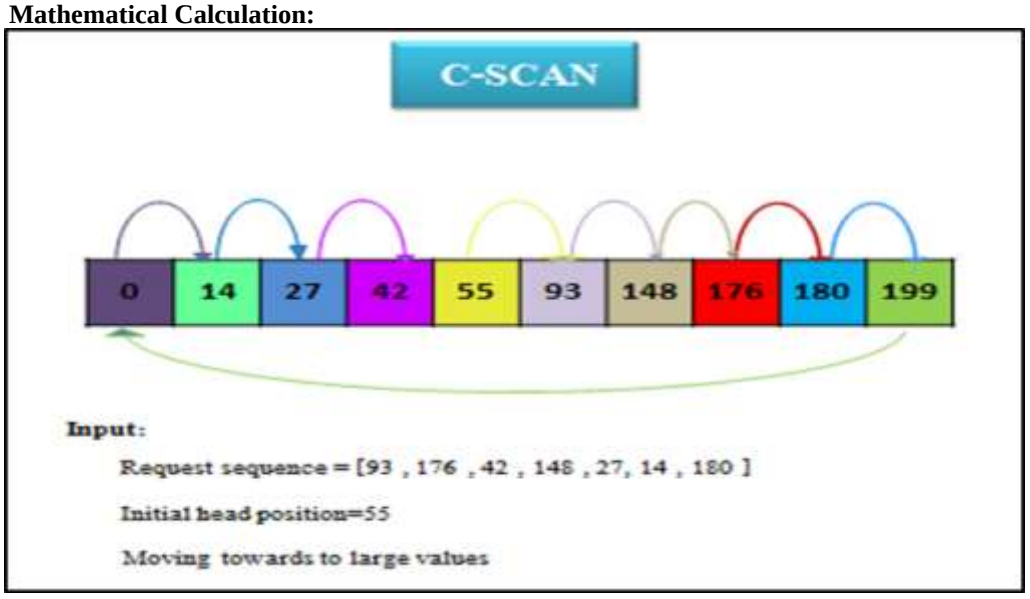

Figure 7: shows inputs of the C-SCAN algorithm

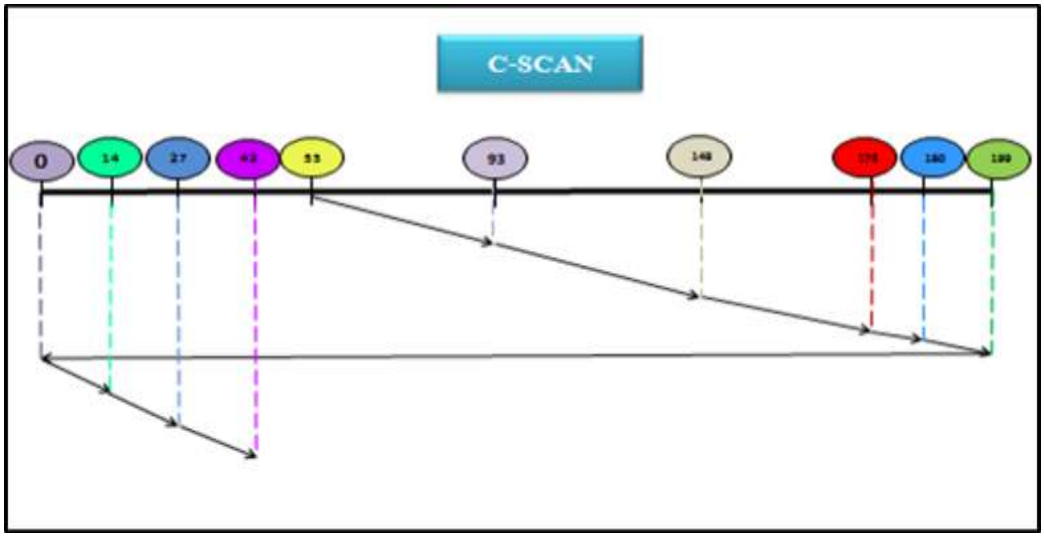

Figure 8: The chart shows the sequence in which requested tracks are serviced using C-SCAN.

The total seek count is calculated as:
(93-55)+(148-93)+(176-148)+(180-176)+(199-180)+(199-0)+(14-0)+(27-14)+(42-27)
=385

2.5 LOOK Disk Scheduling

The LOOK algorithm is an optimized, intelligent version of SCAN. Instead of forcing the disk arm to travel blindly to the physical boundaries of the disk platter, it scans ahead and reverses its direction immediately after servicing the furthest request in the current direction (Silberschatz et al., 2018).

- **Advantages:** Enhances performance compared to SCAN by eliminating unproductive and redundant movements to the outer edges (Kumar & Rajendra, 2015).
- **Disadvantages:** Can still lead to slightly prolonged waiting times for tracks that were just missed or bypassed prior to direction reversal(Yashvir & Prakash, 2012).

- **Mathematical Calculation:**

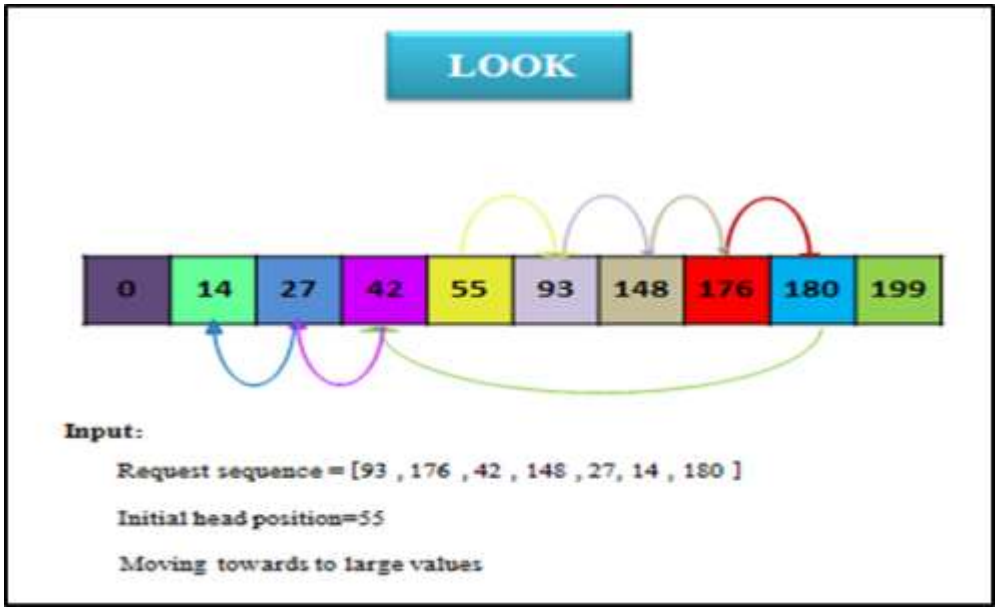

Figure 9: shows inputs of the LOOK algorithm

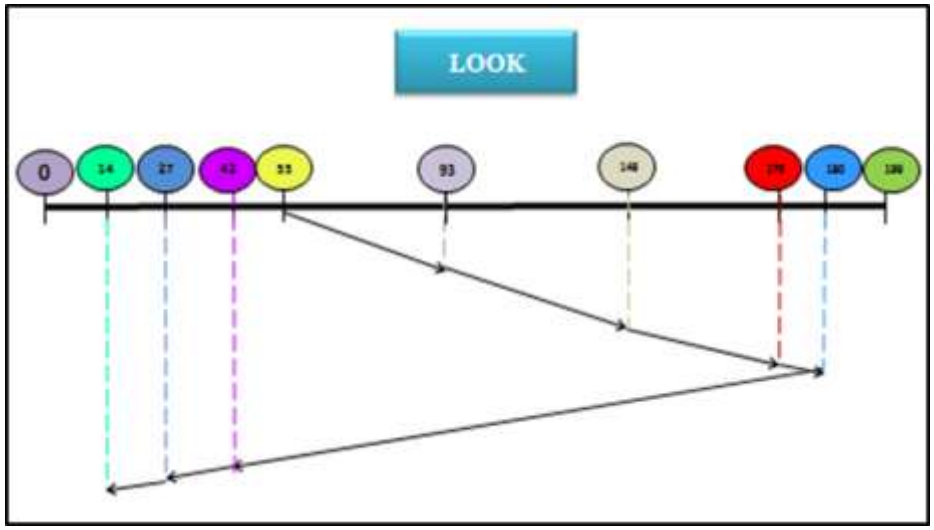

Figure 10: The chart shows the sequence in which requested tracks are serviced using LOOK.

The total seek count is calculated as: Output:
(93-55)+(148-93)+(176-148)+(180-176)+(180-42)+(42-27)+(27-14) =291

2.6 Circular LOOK (C-LOOK)

C-LOOK merges the structural concepts of both C-SCAN and LOOK. The head progresses in a single direction, fulfilling requests until it satisfies the final request in that direction. It then directly teleports back to the furthest outstanding request on the opposite side of the queue, bypassing the physical disk edges entirely (Silberschatz et al., 2018).

- **Advantages:** Exceptional seek optimization that eliminates edge-travel overhead while maintaining a balanced, uniform response time(Kumar & Rajendra, 2015).
- **Disadvantages:** Demands continuous algorithmic overhead to track and verify the maximum and minimum boundaries of the active request queue(Yashvir & Prakash, 2012).

- **Mathematical Calculation:**

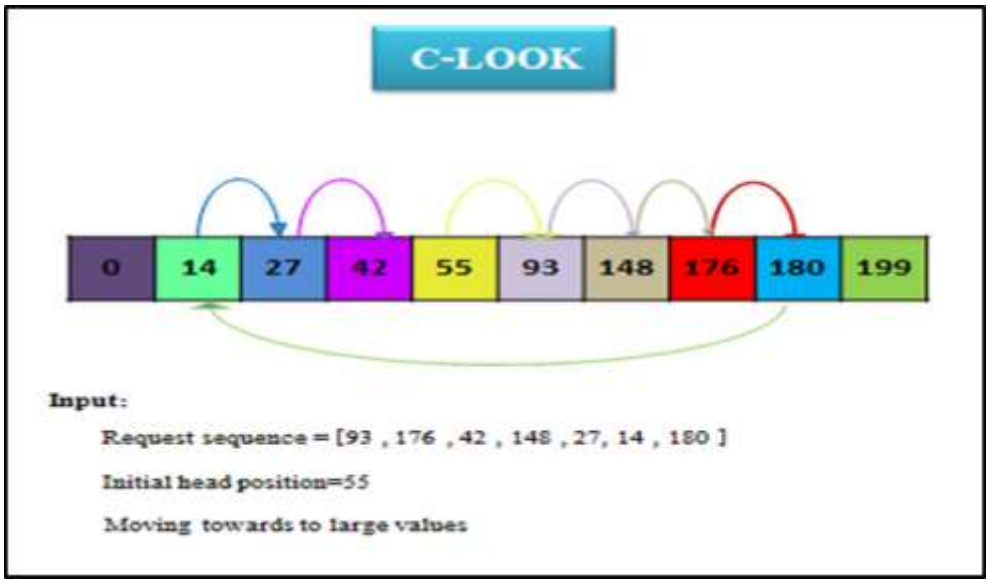

Figure 11: shows inputs of the C-LOOK algorithm

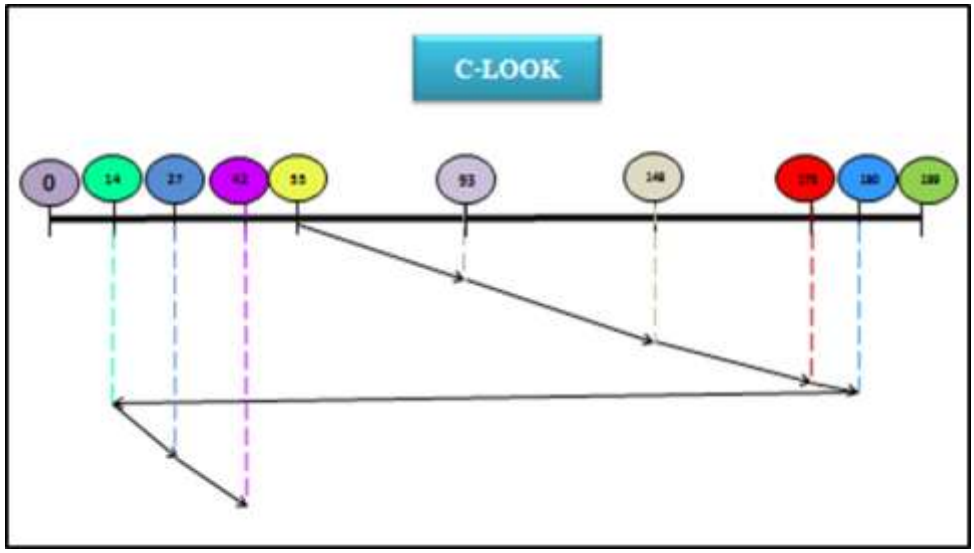

Figure 12: The chart shows the sequence in which requested tracks are serviced using LOOK.

The total seek count is calculated as:

(93-55)+(148-93)+(176-148)+(180-176)+(180-14)+(27-14)+(42-27) =319

3. Results and discussion

By compiling and scrutinizing the data gathered from the controlled case study, the behavior of each algorithm can be rigorously benchmarked. Table 1 outlines the qualitative operational trade-offs concerning starvation risk, throughput, and response latency.

Table 1: Qualitative Comparison of Operational Characteristics

Algorithm	Starvation Vulnerability	System Throughput	Average Response Time
FCFS	No	Low	High
SSTF	Yes	High	Low
SCAN	No	High	Low
C-SCAN	No	High	Low
LOOK	No	High	Low
C-LOOK	No	High	Low

Furthermore, Table 2 delivers the exact numerical results for the Total Head Movement (THM) under the fixed initial head position of track 55, showcasing the true performance divergence of the mechanical drive arm:

Table 2: Quantitative Performance Comparison (Total Seek Tracks)

Algorithm	Initial Head Position	Total Head Movement (Tracks)	Performance Index (Relative to FCFS)
FCFS	55	661 tracks	Baseline (100%)
SSTF	55	**207 tracks**	Optimized (~31.3%)
SCAN	55	329 tracks	Balanced (~49.7%)
C-SCAN	55	385 tracks	Uniform (~58.2%)
LOOK	55	**291 tracks**	High Efficiency (~44.0%)
C-LOOK	55	319 tracks	Uniform/Efficient (~48.2%)

Discussion:

The empirical data indicates that SSTF yields the absolute minimal head movement (207 tracks), proving highly effective for localized queue segments. However, its inherent starvation risk renders it problematic for unpredictable, heavy multi-user workloads. Conversely, the look-ahead class of algorithms (LOOK and C-LOOK) delivers an exceptional operational sweet spot. By registering lower track distances (291 and 319 tracks respectively) compared to standard SCAN/C-SCAN and eliminating edge overhead entirely, they achieve structural equilibrium between maximizing physical mechanical efficiency and guaranteeing request execution fairness.

Conclusion

The architectural reliability and response efficiency of an operating system are profoundly tied to the physical behavior of its secondary storage subsystems. This comparative study demonstrates that choosing a disk scheduling strategy requires a careful balance between raw seek minimization and starvation avoidance. Based on our quantitative findings, this paper recommends the adoption of look-ahead paradigms (**LOOK** and **C-LOOK**) as standard implementations in general-purpose computing environments. These algorithms successfully discard the idle mechanical movements associated with edge-to-edge scanning, ensuring optimal hardware lifespan protection and high-throughput execution

References

[1] GeeksforGeeks. (n.d.). Disk scheduling algorithms. https://www.geeksforgeeks.org/disk-scheduling-algorithms/

[2] Geist, R., & Daniel, S. (1987). A continuum of disk scheduling algorithms. ACM Transactions on Computer Systems, 5(1), 77–92. https://doi.org/10.1145/7351.8292

[3] Jacobson, D. M., & Wilkes, J. (1991). Disk scheduling algorithms based on rotational position (Technical Report No. HPL-CSP-91-7rev1). Hewlett-Packard Laboratories.

[4] Javed, M. Y., & Khan, I. U. (2000). Simulation and performance comparison of four disk scheduling algorithms. In IEEE Region 10 Annual International Conference Proceedings/TENCON (Vol. 2, pp. 10–15). https://doi.org/10.1109/TENCON.2000.888379

[5] Kumar, M. R. M., & Rajendra, B. R. (2015). An improved approach to maximize the performance of disk scheduling algorithm by minimizing the head movement and seek time using Sort Mid Current Comparison (SMCC) algorithm. Procedia Computer Science, 5, 222–231. https://doi.org/10.1016/j.procs.2015.07.468

[6] Paida, H. (2013). Disk scheduling. International Journal of Computer Trends and Technology, 1(2), 6–9.

[7] Silberschatz, A., Galvin, P. B., & Gagne, G. (2018). Operating system concepts (10th ed.). Wiley.

[8] Yashvir, P. S., & Prakash, O. (2012). Selection of scheduling algorithm. International Journal of Engineering and Computer Science, 1(2), 1–9.