Analisis Kinerja Algoritma Penjadwalan CPU Menggunakan CPU Scheduling Simulator Natnuts

Anastasia Naila[1], Akhsani Taqwiym[2]
[1]Manajemen Informasi, Institut Teknologi dan Bisnis Bina Sriwijaya, Palembang, Indonesia
[2]Sistem Informasi, Institut Teknologi dan Bisnis Bina Sriwijaya, Palembang, Indonesia

Article Info

Article history:

Received Dec 16, 2025
Revised Feb 16, 2026
Accepted Feb 16, 2026

Keywords:

CPU Scheduling;
FCFS;
Komparatif;
Priority;
Simulasi;
SJF.

ABSTRACT

Penelitian ini menganalisis kinerja algoritma penjadwalan CPU *First Come First Serve* (FCFS), *Shortest Job First* (SJF), dan Priority melalui pendekatan simulasi komparatif menggunakan CPU *Scheduling* Simulator by Natnuts. Evaluasi dilakukan pada empat skenario pengujian yang mencakup variasi arrival time, jumlah proses, serta distribusi burst time dan prioritas untuk menguji konsistensi performa pada workload statis maupun dinamis. Indikator kinerja yang digunakan meliputi *Average Waiting Time* (AWT) dan *Average Turnaround Time* (ATAT). Hasil eksperimen menunjukkan bahwa algoritma SJF secara konsisten menghasilkan nilai AWT dan ATAT terendah pada seluruh skenario, dengan peningkatan performa yang semakin signifikan pada workload heterogen dan jumlah proses lebih banyak. FCFS menunjukkan kinerja paling sederhana namun kurang adaptif terhadap variasi durasi proses, terutama pada pola kedatangan bertahap. Algoritma Priority memberikan performa menengah yang sangat bergantung pada distribusi nilai prioritas dan berpotensi menimbulkan ketimpangan layanan. Temuan ini menegaskan bahwa efektivitas algoritma penjadwalan bersifat kontekstual terhadap karakteristik beban kerja. Oleh karena itu, pemilihan algoritma sebaiknya mempertimbangkan tujuan sistem, kompleksitas proses, serta kebutuhan responsivitas dan keadilan layanan.

Corresponding Author:

Anastasia Naila
Manajemen Informasi,
Institut Teknologi dan Bisnis Bina Sriwijaya, Palembang, Indonesia,
Email: anastasianaila14@gmail.com.

How to Cite: Anastasia Naila and Akhsani Taqwiym, "Analisis Kinerja Algoritma Penjadwalan CPU Menggunakan CPU Scheduling Simulator Natnuts," Journal of Logic, Informatics, and Algorithm, vol. 1, no. 1, pp. 10–16, Feb. 2026.

1 PENDAHULUAN

Sistem operasi berperan penting dalam mengelola sumber daya perangkat keras, termasuk prosesor, agar eksekusi berbagai proses dapat berlangsung stabil dan efisien. Pada lingkungan komputasi modern, banyak proses dapat berjalan secara bersamaan sehingga diperlukan mekanisme penjadwalan CPU untuk menentukan urutan proses yang memperoleh akses ke prosesor. Keputusan penjadwalan ini memengaruhi indikator kinerja utama, seperti waiting time, turnaround time, response time, dan throughput, yang pada akhirnya menentukan responsivitas sistem secara keseluruhan [15].

Beragam algoritma penjadwalan CPU telah dikembangkan dan digunakan sesuai karakteristik kebutuhan sistem. Algoritma First Come First Serve (FCFS) dikenal sederhana dan mudah diimplementasikan, namun dapat menimbulkan convoy effect ketika proses berdurasi panjang menahan proses lain [5]. Algoritma Shortest Job First (SJF) cenderung menghasilkan waktu tunggu lebih kecil, tetapi menuntut informasi atau prediksi durasi burst time sehingga penerapannya tidak selalu mudah pada kondisi nyata [5], [7]. Sementara itu, algoritma Priority efektif untuk kebutuhan sistem yang menekankan tingkat kepentingan proses, namun berpotensi menimbulkan starvation pada proses berprioritas rendah apabila tidak disertai mekanisme penyeimbang [5], [14]. Sejumlah penelitian komparatif dan tinjauan literatur juga menegaskan bahwa perbedaan performa algoritma sangat dipengaruhi oleh pola kedatangan proses, variasi burst time, dan kebijakan prioritas yang digunakan [5], [6], [12].

Meskipun studi terkait perbandingan penjadwalan CPU sudah banyak dilakukan, masih terdapat kebutuhan untuk menyajikan evaluasi yang lebih terukur, sederhana, dan mudah direplikasi pada skenario proses tertentu, terutama untuk konteks pembelajaran dan praktik analisis kinerja sistem operasi. Banyak studi membandingkan algoritma pada skenario yang beragam, namun sering kali fokus pada kombinasi algoritma tertentu (misalnya FCFS vs Round Robin atau SJF vs Round Robin), sehingga perbandingan FCFS–SJF–Priority pada satu skenario yang sama dan dievaluasi dengan metrik yang konsisten tetap relevan sebagai rujukan praktis [5], [6], [9], [12]. Selain itu, pendekatan simulasi merupakan cara yang umum digunakan untuk mengevaluasi sistem secara terkontrol, karena memungkinkan perbandingan metode secara adil berdasarkan parameter input yang sama [2].

Berdasarkan gap tersebut, penelitian ini bertujuan untuk membandingkan kinerja algoritma FCFS, SJF, dan *Priority* menggunakan pendekatan simulasi komparatif, dengan indikator evaluasi berupa average waiting time, turnaround time, response time, dan throughput [5], [15]. Simulasi dilakukan menggunakan himpunan proses dengan variasi burst time P1=8, P2=5, P3=10 (arrival time seluruhnya 0), serta nilai prioritas P1=3, P2=4, dan P3=6. Fokus penelitian diarahkan untuk mengidentifikasi pola keunggulan masing-masing algoritma pada skenario tersebut sehingga hasilnya dapat dipakai sebagai dasar pertimbangan pemilihan algoritma sesuai karakteristik beban kerja.

Kontribusi penelitian ini dinyatakan secara eksplisit sebagai berikut: (1) menyediakan perbandingan terukur dan konsisten antara FCFS, SJF, dan Priority pada skenario proses yang sama dengan metrik evaluasi standar (AWT, TAT, RT, *throughput*) [5], [15]; (2) menyajikan interpretasi keunggulan algoritma yang dikaitkan dengan karakteristik beban kerja (variasi burst time dan prioritas) sehingga hasil tidak sekadar berupa angka [5], [12]; dan (3) memberikan rekomendasi praktis bagi pengelola/pengembang sistem maupun konteks pembelajaran dalam menentukan algoritma penjadwalan CPU yang sesuai kebutuhan, karena tidak ada satu algoritma yang selalu unggul pada semua kondisi [5], [15].

Dengan demikian, penelitian ini diharapkan dapat memperkuat pemahaman mengenai hubungan antara karakteristik proses dan performa algoritma penjadwalan CPU, serta membantu pengguna menentukan strategi penjadwalan yang paling sesuai untuk mencapai efisiensi dan responsivitas sistem yang optimal [15].

2 METODE PENELITIAN

Penelitian ini menggunakan pendekatan simulasi komparatif untuk membandingkan kinerja tiga algoritma penjadwalan CPU, yaitu *First Come First Serve* (FCFS), *Shortest Job First* (SJF), dan *Priority*. Pendekatan simulasi dipilih karena memungkinkan pengujian yang terkontrol dan adil, di mana seluruh algoritma dievaluasi menggunakan parameter input yang sama sehingga perbedaan hasil dapat dikaitkan langsung dengan karakteristik algoritmanya [2], [15]. Evaluasi dilakukan berdasarkan metrik kinerja standar penjadwalan CPU, meliputi average waiting time (AWT), *average turnaround time* (ATAT), *average response time* (ART), serta *throughput* [5], [15].

2.1. Desain Penelitian

Desain penelitian bersifat deskriptif-kuantitatif berbasis simulasi, yaitu menghasilkan keluaran numerik dari setiap algoritma penjadwalan berdasarkan skenario proses yang ditetapkan. Penelitian tidak melakukan manipulasi perangkat keras maupun pengukuran performa mesin secara fisik, melainkan menilai performa algoritma melalui hasil eksekusi simulasi yang merepresentasikan urutan dan waktu layanan proses pada CPU [2], [15].

2.2. Simulasi Penelitian

Simulasi dilakukan menggunakan CPU Scheduling Simulator by Natnuts sebagai alat bantu untuk memvisualisasikan urutan eksekusi proses dan menghitung metrik kinerja penjadwalan. Pemilihan simulator bertujuan mempermudah replikasi serta

memastikan perhitungan metrik dilakukan secara konsisten antar algoritma pada input yang sama [5], [12]. Seluruh pengujian dilakukan pada satu skenario proses yang sama untuk tiap algoritma agar perbandingan bersifat fair.

2.3. Skenario Pengujian

Penelitian ini menggunakan satu set proses yang ditetapkan sebagai skenario uji, yaitu tiga proses dengan arrival time seluruhnya bernilai 0 dan variasi burst time masing-masing P1 = 8, P2 = 5, dan P3 = 10. Selain itu, untuk mendukung pengujian pada algoritma *Priority*, setiap proses diberikan nilai prioritas, yaitu Priority(P1) = 3, *Priority*(P2) = 4, dan Priority(P3) = 6. Skenario ini dipilih untuk memperlihatkan dampak variasi burst time serta kebijakan prioritas terhadap kinerja penjadwalan, khususnya pada waktu tunggu, waktu penyelesaian, respons awal, dan laju penyelesaian proses (*throughput*) [5], [15].

2.4. Variabel dan Metrik Evaluasi

Variabel bebas dalam penelitian ini adalah jenis algoritma penjadwalan CPU, yaitu FCFS, SJF, dan Priority. Adapun variabel terikatnya berupa metrik kinerja penjadwalan yang diperoleh dari hasil simulasi, meliputi *Waiting Time* (WT) yang merepresentasikan waktu tunggu proses di ready queue sebelum dieksekusi, *Turnaround Time* (TAT) yaitu total waktu sejak proses datang hingga selesai (*completion time − arrival time*), serta *Response Time* (RT) yang menunjukkan waktu sejak proses datang sampai pertama kali memperoleh jatah eksekusi CPU. Selain itu, penelitian juga mengukur throughput, yakni jumlah proses yang berhasil diselesaikan per satuan waktu (jumlah proses selesai dibagi total waktu penyelesaian). Nilai *average* (rata-rata) untuk WT, TAT, dan RT dihitung dengan menjumlahkan nilai masing-masing proses kemudian membaginya dengan jumlah proses yang diuji [5], [15].

2.4. Simulasi

Langkah-langkah simulasi dalam penelitian ini diawali dengan menetapkan data proses P1, P2, dan P3 beserta parameter yang dibutuhkan, yaitu arrival time, burst time, serta nilai prioritas (khusus untuk algoritma *Priority*). Setelah data proses ditetapkan, simulasi pertama dijalankan menggunakan algoritma FCFS dengan mencatat urutan eksekusi proses serta hasil metrik WT, TAT, RT, dan throughput. Selanjutnya, dengan menggunakan input proses yang sama, simulasi dijalankan kembali untuk algoritma SJF dan seluruh metrik yang sama dicatat. Prosedur yang sama juga diterapkan pada algoritma Priority, yaitu menjalankan simulasi dengan input identik dan mencatat metrik kinerja yang dihasilkan. Untuk memastikan perbandingan antar algoritma bersifat adil dan valid, penelitian ini menjaga agar seluruh parameter selain jenis algoritma termasuk jumlah proses, *arrival time*, *burst time*, dan prioritas tetap sama sehingga perbedaan hasil benar-benar mencerminkan karakteristik algoritma penjadwalan yang diuji [2], [5].

2.5. Teknik Analisis Data

Analisis data dalam penelitian ini dilakukan secara deskriptif-komparatif. Hasil metrik dari masing-masing algoritma disajikan dalam tabel perbandingan untuk mengidentifikasi: (1) algoritma dengan *Average Waiting Time* (AWT) terendah, (2) perubahan *Average Turnaround Time* (ATAT) dan *Average Response Time* (ART) antar algoritma, serta (3) perbedaan throughput pada skenario pengujian yang sama. Selanjutnya, dilakukan interpretasi berdasarkan karakteristik masing-masing algoritma untuk menjelaskan mengapa suatu algoritma unggul pada metrik tertentu, sekaligus menegaskan bahwa tidak ada algoritma yang selalu optimal pada semua kondisi beban kerja [5], [15].

2.6. Evaluasi Pembelajaran

Selain simulasi komparatif, penelitian dapat dilengkapi dengan evaluasi pembelajaran menggunakan desain *one-group pretest–posttest* untuk mengukur perubahan pemahaman mahasiswa setelah praktikum menggunakan simulator [3]. Skor pretest dan posttest dianalisis secara deskriptif (*mean, median, min–maks*) dan dibandingkan untuk melihat kecenderungan peningkatan. Namun, bagian ini diposisikan sebagai tujuan sekunder (pendukung pembelajaran), terpisah dari tujuan utama komparasi kinerja algoritma [3], [5].

2.7. Skenario Eksperimen

Pada penelitian ini, pengujian dilakukan melalui empat skenario eksperimen untuk memperoleh gambaran performa algoritma secara lebih komprehensif. Skenario pertama (*baseline*) menggunakan tiga proses dengan arrival time masing-masing 0, 0, dan 0; burst time sebesar 8, 5, dan 10; serta nilai prioritas 3, 4, dan 6. Skenario ini merepresentasikan kondisi statis di mana seluruh proses datang secara bersamaan sehingga perbedaan kinerja algoritma sepenuhnya dipengaruhi oleh mekanisme penjadwalan dan urutan eksekusi berdasarkan burst time maupun prioritas.

Selanjutnya, Skenario kedua dirancang dengan pola arrival time yang berbeda, yaitu 0, 2, dan 4, sementara *burst time* dan nilai prioritas tetap sama seperti skenario pertama (8, 5, 10 dan 3, 4, 6). Skenario ini bertujuan untuk menguji pengaruh variasi waktu kedatangan proses terhadap perubahan waiting time dan response time, serta untuk melihat bagaimana algoritma merespons dinamika antrian ketika proses tidak datang secara simultan.

Skenario ketiga meningkatkan kompleksitas dengan menambah jumlah proses menjadi lima, seluruhnya memiliki *arrival time 0*, dengan variasi *burst time* 3, 7, 2, 9, dan 5 serta nilai prioritas 2, 1, 4, 3, dan 5. Skenario ini digunakan untuk menguji

performa algoritma pada *workload* yang lebih kompleks dan heterogen, sehingga dapat diamati konsistensi performa masing-masing algoritma ketika beban kerja meningkat.

Terakhir, Skenario keempat mensimulasikan workload campuran dan semi-dinamis dengan *arrival time* 0, 1, 3, 5, dan 6; *burst time* 12, 4, 7, 2, dan 9; serta prioritas 5, 2, 4, 1, dan 3. Kombinasi ini merepresentasikan kondisi sistem yang lebih realistis dengan variasi proses pendek dan panjang yang datang secara bertahap. Tujuan dari skenario ini adalah untuk menganalisis respons algoritma terhadap lingkungan yang lebih dinamis serta mengamati potensi ketimpangan layanan, seperti peningkatan waiting time atau risiko starvation pada algoritma tertentu.

2.8. Analisis Kompleksitas Algoritma

Selain evaluasi metrik performa, penelitian ini juga membandingkan kompleksitas waktu teoritis dari masing-masing algoritma dapat dilihat pada tabel 1.

Tabel 1. Kompleksitas Algoritma

Algoritma	Mekanisme	Time Complexity
FCFS	Antrian FIFO	O(n)
SJF	Sorting burst time	O(n log n)
Priority	Sorting berdasarkan prioritas	O(n log n)

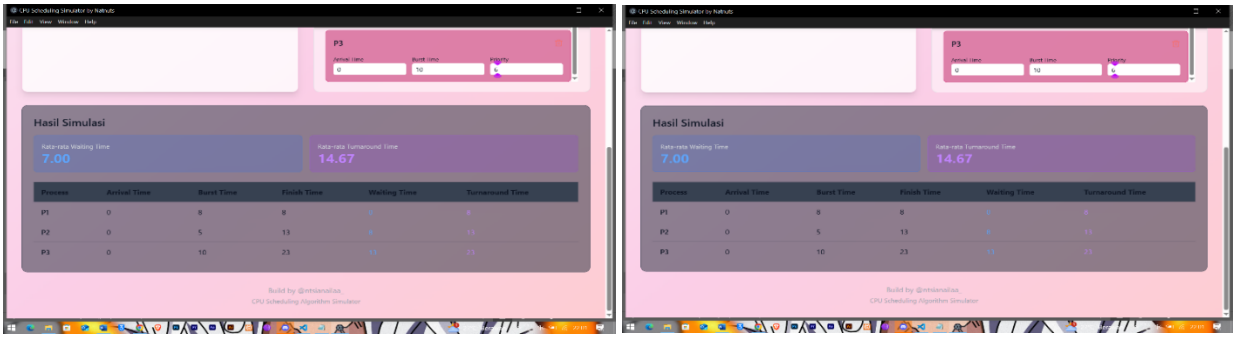
Meskipun FCFS memiliki kompleksitas paling sederhana, hasil simulasi menunjukkan bahwa kompleksitas rendah tidak selalu menghasilkan metrik performa terbaik.

3 HASIL DAN PEMBAHASAN

3.1. Hasil Simulasi Penjadwalan CPU

Simulasi dilakukan pada tiga proses dengan arrival time seluruhnya 0 dan burst time masing-masing P1=8, P2=5, dan P3=10. Evaluasi kinerja algoritma menggunakan metrik *Average Waiting Time* (AWT), *Average Turnaround Time* (ATAT), *Average Response Time* (ART), serta throughput. Karena seluruh proses datang pada waktu yang sama dan algoritma yang digunakan bersifat non-preemptive, maka ART bernilai sama dengan AWT (response time setara dengan waktu tunggu sebelum pertama kali memperoleh CPU). Throughput dihitung sebagai jumlah proses selesai dibagi total waktu penyelesaian.

Berdasarkan hasil simulasi, algoritma FCFS menghasilkan AWT = 7,00, ATAT = 14,67, ART = 7,00, dan throughput 0,13 proses per satuan waktu. Pada algoritma SJF, proses dengan burst time terpendek dieksekusi lebih dahulu sehingga metrik membaik menjadi AWT = 6,00, ATAT = 13,67, ART = 6,00, dengan throughput tetap 0,13. Untuk algoritma *Priority* (dengan asumsi kebijakan prioritas pada simulator tidak mengubah urutan dari kasus FCFS pada skenario ini), nilai metrik yang diperoleh sama dengan FCFS, yaitu AWT = 7,00, ATAT = 14,67, ART = 7,00, dan throughput 0,13.



Gambar 1. Percobaan FCFS dan SJF.

Tabel 2. Komparasi Kinerja FCFS, SJF dan Priority

Algoritma	AWT	ATAT	ART	Throughput
FCFS	7,00	14,67	7,00	0,13
SJF (non-preemptive)	6,00	13,67	6,00	0,13
Priority (non-preemptive)*	7,00	14,67	7,00	0,13

Hasil Priority bergantung pada aturan prioritas (apakah angka kecil/angka besar dianggap lebih tinggi) dan nilai prioritas yang digunakan. Jika kebijakan prioritas mengubah urutan eksekusi, maka AWT/ATAT/ART dapat berubah, sedangkan throughput cenderung tetap sama karena total burst time tidak berubah

3.2. Pembahasan Komparatif (Trade-off)

Hasil komparasi menunjukkan bahwa perbedaan kinerja utama antar algoritma tampak pada AWT/ART dan ATAT, sedangkan throughput relatif sama. Hal ini terjadi karena pada skenario uji, total beban kerja CPU (jumlah *burst time*) tetap, sehingga algoritma hanya mengubah urutan eksekusi, bukan mengubah total waktu komputasi yang dibutuhkan sistem.

Algoritma SJF memberikan hasil paling efisien pada skenario ini (AWT/ART paling rendah dan ATAT lebih kecil) karena mendahulukan proses dengan burst time terpendek. Dampaknya, proses pendek tidak tertahan oleh proses panjang yang dieksekusi lebih awal, sehingga waktu tunggu rata-rata menurun. Namun, trade-off SJF adalah kebutuhan akan informasi/estimasi burst time; pada sistem nyata, informasi ini sering tidak tersedia secara pasti. Selain itu, pada kondisi kedatangan proses yang dinamis, SJF dapat menunda proses panjang dalam waktu lama apabila proses pendek terus berdatangan (potensi ketidakadilan pada kasus tertentu).

Algoritma FCFS unggul dari sisi kesederhanaan dan kemudahan implementasi karena melayani proses sesuai urutan kedatangan. Akan tetapi, FCFS rentan menghasilkan waktu tunggu lebih besar ketika proses berdurasi panjang berada di depan antrean, karena proses berikutnya harus menunggu hingga proses tersebut selesai. Kondisi ini berkaitan dengan fenomena convoy effect, yang membuat FCFS kurang optimal untuk sistem yang menuntut respons cepat.

Algoritma Priority dirancang untuk mendahulukan proses dengan tingkat kepentingan lebih tinggi. Keunggulannya adalah mampu mempercepat layanan untuk proses kritis, tetapi *trade-off* utamanya adalah risiko starvation pada proses berprioritas rendah apabila tidak ada mekanisme penyeimbang. Pada skenario uji ini, hasil Priority dapat sama dengan FCFS apabila urutan prioritas yang diterapkan tidak mengubah urutan eksekusi proses; sebaliknya, jika urutan prioritas berbeda, metrik AWT/ATAT/ART dapat berubah sesuai urutan eksekusi yang terbentuk.

3.3. Rekomendasi Pemilihan Algoritma Berdasarkan Karakteristik *Workload*

Berdasarkan hasil dan trade-off yang diperoleh, rekomendasi pemilihan algoritma dapat dirumuskan sebagai berikut: SJF lebih sesuai untuk *workload* yang menuntut respons cepat dan didominasi proses berdurasi pendek karena mampu menurunkan rata-rata waktu tunggu dan waktu respons; FCFS dapat digunakan pada *workload* yang sederhana dengan kebutuhan implementasi yang mudah serta beban kerja yang relatif stabil karena mekanismenya sederhana dan mudah dipahami; sedangkan Priority lebih tepat untuk *workload* yang memerlukan perlakuan khusus terhadap proses tertentu, namun perlu disertai mekanisme pengendalian agar tidak terjadi starvation pada proses berprioritas rendah. Secara umum, temuan ini menegaskan bahwa tidak ada algoritma yang selalu unggul pada semua kondisi, sehingga pemilihan algoritma penjadwalan CPU sebaiknya disesuaikan dengan tujuan sistem (*responsivitas*, *fairness*, atau prioritas) serta karakteristik beban kerja, seperti variasi burst time dan kebutuhan penanganan proses kritis.

3.4. Komparasi Multi-Skenario

Berdasarkan hasil rekapitulasi pada Tabel 3 dan Tabel 4, terlihat adanya pola konsistensi performa antar algoritma pada empat skenario pengujian yang berbeda.

Tabel 3. Rekapitulasi *Average Waiting Time* (AWT)

Skenario	FCFS	SJF	Priority
1	7.00	6.00	7.00
2	6.67	5.33	6.33
3	8.20	4.80	6.40
4	10.40	5.20	7.00

Tabel 4. Rekapitulasi *Average Turnaround Time* (ATAT)

Skenario	FCFS	SJF	Priority
1	14.67	13.67	14.67
2	14.00	12.67	13.33
3	13.80	10.40	12.20
4	17.60	12.40	14.60

Pada metrik *Average Waiting Time* (AWT), algoritma SJF secara konsisten menghasilkan nilai terendah pada seluruh skenario, yaitu 6,00 pada skenario 1; 5,33 pada skenario 2; 4,80 pada skenario 3; dan 5,20 pada skenario 4. Sebaliknya, FCFS menunjukkan kecenderungan nilai AWT yang lebih tinggi, terutama ketika kompleksitas workload meningkat, dengan nilai tertinggi mencapai 10,40 pada skenario 4. Algoritma Priority berada di antara keduanya dengan performa yang fluktuatif, yaitu 7,00; 6,33; 6,40; dan 7,00 pada masing-masing skenario.

Pola serupa juga terlihat pada metrik *Average Turnaround Time* (ATAT). Algoritma SJF kembali menunjukkan performa terbaik dengan nilai terendah secara konsisten (13,67; 12,67; 10,40; dan 12,40). FCFS menghasilkan ATAT yang relatif lebih tinggi, khususnya pada skenario 4 dengan nilai 17,60, yang menunjukkan peningkatan signifikan ketika proses datang secara bertahap dan memiliki variasi burst time yang besar. Sementara itu, algoritma *Priority* memberikan nilai menengah (14,67; 13,33; 12,20; dan 14,60), yang menunjukkan bahwa efektivitasnya sangat dipengaruhi oleh distribusi nilai prioritas pada masing-masing skenario.

Secara umum, hasil komparasi multi-skenario ini menunjukkan bahwa SJF memiliki konsistensi performa terbaik dalam menekan waktu tunggu dan waktu penyelesaian rata-rata, terutama pada workload heterogen dengan variasi burst time yang signifikan. Perbedaan performa antar algoritma semakin terlihat pada skenario dengan jumlah proses lebih banyak dan pola kedatangan dinamis (skenario 3 dan 4), di mana selisih AWT dan ATAT antara FCFS dan SJF menjadi semakin besar. Hal ini mengindikasikan bahwa kompleksitas dan dinamika *workload* memperbesar dampak mekanisme pemilihan proses pada masing-masing algoritma.

Temuan ini memperkuat argumentasi bahwa keunggulan algoritma penjadwalan CPU bersifat kontekstual terhadap karakteristik beban kerja. Pada sistem dengan variasi durasi proses yang tinggi, algoritma berbasis pemilihan durasi terpendek seperti SJF lebih efektif dalam meningkatkan efisiensi waktu tunggu dan penyelesaian proses dibandingkan pendekatan antrian sederhana seperti FCFS.

4 KESIMPULAN

Berdasarkan hasil simulasi komparatif pada empat skenario pengujian yang mencakup variasi arrival time, jumlah proses, serta distribusi burst time dan prioritas, dapat disimpulkan bahwa algoritma penjadwalan CPU menunjukkan performa yang berbeda secara signifikan tergantung pada karakteristik beban kerja. Algoritma *Shortest Job First* (SJF) secara konsisten menghasilkan nilai *Average Waiting Time* (AWT) dan *Average Turnaround Time* (ATAT) terendah pada seluruh skenario, menunjukkan kemampuannya dalam mengoptimalkan efisiensi waktu tunggu dan waktu penyelesaian proses, khususnya pada *workload* yang heterogen dan dinamis.

Sebaliknya, algoritma *First Come First Serve* (FCFS) cenderung menghasilkan waktu tunggu dan turnaround time yang lebih tinggi, terutama ketika jumlah proses bertambah dan pola kedatangan tidak simultan. Hal ini menunjukkan bahwa mekanisme antrian sederhana kurang adaptif terhadap variasi durasi proses dan dinamika sistem. Algoritma *Priority* memberikan performa yang berada di antara FCFS dan SJF, dengan efektivitas yang sangat dipengaruhi oleh distribusi nilai prioritas yang digunakan. Pada skenario tertentu, algoritma ini mampu meningkatkan respons terhadap proses kritis, namun tetap memiliki potensi ketimpangan layanan apabila tidak disertai mekanisme pengendalian seperti aging.

Hasil eksperimen multi-skenario ini menegaskan bahwa tidak terdapat algoritma yang bersifat universal terbaik untuk semua kondisi sistem. Pemilihan algoritma penjadwalan sebaiknya mempertimbangkan tujuan utama sistem, apakah berfokus pada minimisasi waktu tunggu, pemerataan layanan, atau penanganan proses prioritas. Secara umum, SJF lebih direkomendasikan untuk lingkungan dengan variasi durasi proses yang tinggi dan kebutuhan respons cepat, FCFS sesuai untuk sistem sederhana dengan beban stabil, sedangkan *Priority* tepat digunakan pada sistem yang menuntut perlakuan khusus terhadap proses tertentu dengan tetap memperhatikan aspek keadilan.

Dengan demikian, penelitian ini memberikan dasar empiris yang lebih robust melalui evaluasi multi-skenario dalam mendukung pengambilan keputusan pemilihan algoritma penjadwalan CPU yang sesuai dengan karakteristik workload dan kebutuhan sistem..

5 DECLARATIONS

AI USAGE STATEMENT

Selama penyusunan karya ini, penulis menggunakan ChatGPT (OpenAI) sebagai alat bantu untuk meningkatkan kualitas bahasa, kejelasan penyampaian, dan kerapian struktur naskah. Setelah menggunakan alat tersebut, penulis meninjau kembali seluruh keluaran, melakukan penyuntingan seperlunya, serta memastikan kesesuaian isi dengan data dan tujuan penelitian. Penulis bertanggung jawab sepenuhnya atas seluruh konten, interpretasi, dan kesimpulan yang tercantum dalam publikasi ini.

REFERENCES

[1] K. Marzuki, A. Setyanto, and A. Nasiri, "Audit Tata Kelola Teknologi Informasi Menggunakan COBIT 4.1 Domain Monitoring Evaluasi pada Perguruan Tinggi Swasta," BITe: Journal Bumigora Information Technology, vol. 3, no. 3, pp. 1–10, 2021

[2] P. M. Morse and H. Feshback, *Methods of Theoretical Physics*. New York: McGraw Hill, 1953.

[3] P. S. Meszaros, S. Lee, and A. Laughlin, "Information processing and information technology career interest and choice among high school students," *Reconfiguring the Firewall*, Wellesley: A K Peters, 2007, pp. 77–86.

[4] A. Author et al., "Analisis Algoritma Penjadwalan Hard Disk Tradisional: Tinjauan," *Jurnal Penelitian Quest*, no

Sinta, 2021.

[5] S. Author et al., “Systematic Literature Review: Perbandingan Kinerja Algoritma Penjadwalan CPU FCFS, SJF, Round Robin, dan Priority,” *Jurnal Ilmiah Sistem Informasi dan Ilmu Komputer*, Sinta 5, 2025.

[6] R. Author et al., “Analisis Perbandingan Algoritma Penjadwalan CPU First Come First Serve dan Round Robin,” *Building of Informatics, Technology and Science (BITS)*, 2021.

[7] Y. Author et al., “Analisis Average Waiting Time Penjadwalan CPU Menggunakan Algoritma Shortest Remaining First dan Round Robin,” *JDMIS: Journal of Data Mining and Information Systems*, 2025.

[8] D. Author et al., “Analisis Penjadwalan Produksi Menggunakan Metode FCFS, SPT, dan EDD pada B21 Digital Printing,” *Repeater: Publikasi Teknik Informatika dan Jaringan*, Sinta 5, 2025.

[9] H. Author et al., “Systematic Literature Review: Perbandingan Algoritma Round Robin dan Shortest Job First dalam Penjadwalan CPU,” *BIOS: Jurnal Teknologi Informasi dan Rekayasa Komputer*, Sinta 5, 2025

[10] S. Author et al., “Perbandingan Kinerja dan Efektivitas Algoritma FCFS dan SJF pada Sistem Antrian UMKM,” *Progresif: Jurnal Ilmiah Komputer*, Sinta 4, 2025.

[11] A. Author et al., “Implementasi Algoritma Shortest Job First pada Sistem Antrean Pelayanan Administrasi Desa,” *JPTI – Jurnal Pendidikan dan Teknologi Indonesia*, Sinta 3, 2025.

[12] R. Author et al., “Analisis Perbandingan Algoritma Penjadwalan Round Robin dan Shortest Job First untuk Manajemen Proses dalam Single Processing,” *semanTIK*, vol. 7, 2021.

[13] G. Author et al., “Implementasi Algoritma Shortest Job First pada Sistem Pembelian Menu di Warkop Geidar,” *J-Ilkominfo*, Sinta 4, 2025.

[14] W. Wisnu, M. Desinta, A. Kezia Jazzlyn, “Implementasi Algoritma Round Robin dan Priority Pada Sistem Antrian Rumah Sakit”, Jurnal Fasilkom, Vol 14, No. 2, PP. 507-513, Agustus 2024.

[15]I. Tasya Fitria., “Analisis Performa Algoritma Penjadwalan CPU Dalam Sistem Operasi Komputer Untuk Pengoptimalan Responsivitas”,Jurnal Teknologi Pintar, Vol. 3, November 2023.