BIOS : Jurnal Teknologi Informasi dan Rekayasa Komputer
Vol. 6, No. 2, September 2025, hlm. 74-83
e-ISSN: 2722-0850

Systematic Literature Review : Perbandingan Algoritma *Round Robin* dan *Shortest Job First* dalam Penjadwalan CPU

Ibnu Fadhilah[1], Herbert Siregar[2]
[1]Ilmu Komputer, Universitas Pendidikan Indonesia, *ibnu.fadhil@upi.edu*
[2]Ilmu Komputer, Universitas Pendidikan Indonesia, *herbert@upi.edu*

Keywords

CPU Scheduling,
Round Robin,
Shortest Job First,
Systematic Literature Review,
Algorithm Performance

ABSTRACT

The Central Processing Unit (CPU) scheduling is a crucial aspect of operating systems that significantly impacts overall system efficiency. Round Robin (RR) and Shortest Job First (SJF) are two commonly used scheduling algorithms, each with distinct characteristics. Although numerous studies have compared these algorithms, a clear consensus on the superiority of either algorithm across various workload scenarios remains elusive. This systematic literature review aims to analyze and synthesize empirical evidence from prior research (2020-2024) comparing the performance of RR and SJF algorithms. Literature searches on four major databases and study selection using the PRISMA framework with strict inclusion criteria (empirical comparative studies from 2020-2024) yielded eight articles for analysis. The synthesis of results indicates that the SJF algorithm, particularly its preemptive variant (SRTF), consistently excels in efficiency, with an Average Waiting Time (AWT) that can be over 50% lower than RR in some scenarios. Conversely, RR demonstrates superiority in response time and fairness in CPU allocation, though its performance is highly dependent on the time quantum size. The selection of an optimal algorithm is highly contingent upon system performance metric priorities, and these findings provide a foundation for the future development of hybrid algorithms.

Kata Kunci

Penjadwalan CPU,
Round Robin,
Shortest Job First,
Kajian Literatur Sistematis,
Kinerja Algoritma

ABSTRAK

Penjadwalan Prosesor Sentral (CPU) adalah aspek krusial dalam sistem operasi yang mempengaruhi efisiensi sistem secara keseluruhan. Algoritma *Round Robin* (RR) dan *Shortest Job First* (SJF) merupakan dua metode penjadwalan yang umum digunakan, masing-masing dengan karakteristik yang berbeda. Meskipun banyak studi telah membandingkan keduanya, belum ada konsensus yang jelas mengenai superioritas salah satu algoritma dalam berbagai skenario beban kerja. Kajian literatur sistematis ini bertujuan untuk menganalisis dan menyintesis bukti empiris dari penelitian terdahulu (2020-2024) yang membandingkan kinerja algoritma RR dan SJF. Pencarian literatur pada empat database utama dan seleksi studi menggunakan kerangka PRISMA dengan kriteria inklusi yang ketat (studi komparatif empiris periode 2020-2024) menghasilkan delapan artikel untuk dianalisis. Hasil sintesis menunjukkan bahwa algoritma SJF, khususnya varian preemptive (SRTF), secara konsisten unggul dalam efisiensi, dengan Waktu Tunggu Rata-rata (AWT) yang dapat 50% lebih rendah dibandingkan RR pada beberapa skenario. Sebaliknya, RR menunjukkan keunggulan pada waktu respons dan keadilan alokasi CPU, meskipun kinerjanya sangat bergantung pada ukuran *time quantum*. Pemilihan algoritma yang optimal sangat bergantung pada prioritas metrik kinerja sistem, dan temuan ini memberikan landasan bagi pengembangan algoritma hibrida di masa depan.

Korespondensi Penulis:

Ibnu Fadhilah,
Universitas Pendidikan Indonesia, Jl. Dr. Setiabudi No. 229,
Bandung, Jawa Barat, Indonesia, 40154
Telepon : +6281318243501
Email: *ibnu.fadhil@upi.edu*

Submitted : 19-06-2025; Accepted : 21-08-2025;
Published : 03-09-2025

1. PENDAHULUAN

1.1. Latar Belakang

Penjadwalan Prosesor Sentral (CPU) merupakan fungsi fundamental dalam sistem operasi yang bertujuan untuk memaksimalkan utilisasi CPU dan kinerja sistem melalui alokasi proses yang efisien [1], [2], [3]. Berbagai algoritma penjadwalan telah dikembangkan, di antaranya *Round Robin* (RR) dan *Shortest Job First* (SJF) menjadi

dua metode klasik yang paling sering diimplementasikan dan dianalisis [2], [3]. Keduanya mewakili filosofi desain yang berbeda: RR berfokus pada keadilan (*fairness*) dan interaktivitas melalui pembagian waktu (*time-sharing*), sementara SJF dirancang untuk efisiensi optimal dalam metrik waktu tunggu (*waiting time*) dan waktu penyelesaian (*turnaround time*) dengan memprioritaskan proses terpendek [1], [2].

Tingginya minat riset terhadap perbandingan kinerja RR dan SJF telah menghasilkan banyak sekali studi individual [4], [5]. Namun, bukti empiris yang ada saat ini terfragmentasi dan sering kali tidak memberikan kesimpulan yang seragam. Beberapa penelitian bersifat parsial, misalnya hanya berfokus pada analisis varian dari satu algoritma saja [4], sementara yang lain temuannya sangat bergantung pada skenario spesifik seperti pada sistem waktu-nyata (*real-time systems*) [5]. Akibatnya, belum ada konsensus yang terpadu mengenai kondisi operasional di mana masing-masing algoritma menunjukkan keunggulan optimalnya. Fragmentasi bukti ini menciptakan kebutuhan mendesak akan sebuah sintesis yang komprehensif. Untuk menjawab kebutuhan tersebut, metode tinjauan literatur yang paling tepat adalah *Systematic Literature Review* (SLR), karena merupakan pendekatan yang terstruktur, dan transparan untuk mengidentifikasi, mengevaluasi, dan menyintesis semua bukti relevan pada suatu topik penelitian tertentu [6].

Berdasarkan penelusuran awal pada basis data akademik utama, sebuah celah signifikan dalam literatur berhasil diidentifikasi. Hingga saat ini, belum ditemukan adanya SLR yang secara eksklusif berfokus pada perbandingan antara algoritma RR dan SJF. Kajian-kajian yang ada cenderung membahas keduanya hanya sebagai bagian dari analisis perbandingan algoritma yang cakupannya jauh lebih luas [7]. Oleh karena itu, kebaruan (*novelty*) utama dari penelitian ini adalah untuk mengisi celah tersebut dengan melakukan SLR pertama yang didedikasikan secara spesifik untuk topik ini. Tujuan penelitian ini adalah untuk secara sistematis menganalisis dan menyajikan bukti-bukti empiris yang ada, guna memberikan gambaran komprehensif mengenai keunggulan dan kelemahan relatif antara algoritma RR dan SJF dalam berbagai skenario operasional.

1.2. Tujuan Penelitian dan Pertanyaan Penelitian

Oleh karena itu, tujuan dari kajian literatur sistematis ini adalah untuk melakukan analisis mendalam dan sintesis terhadap bukti-bukti empiris dari penelitian-penelitian terdahulu yang membandingkan kinerja algoritma penjadwalan CPU *Round Robin* dan *Shortest Job First*, khususnya yang menggunakan pendekatan simulasi atau eksperimen. Kajian ini bertujuan untuk memberikan gambaran yang lebih jelas mengenai efektivitas relatif kedua algoritma tersebut dalam berbagai kondisi operasional. Masalah utama yang diangkat dalam penelitian ini adalah belum adanya gambaran menyeluruh dan sistematis mengenai keunggulan dan kelemahan algoritma RR dan SJF dalam konteks penjadwalan CPU.

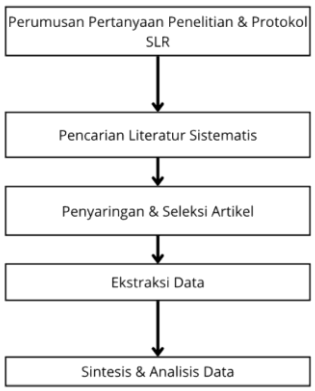
Kajian literatur sistematis ini bertujuan untuk menjawab beberapa pertanyaan penelitian (RQ) yang spesifik: RQ1 berfokus pada perbandingan kinerja algoritma *Round Robin* (RR) dan *Shortest Job First* (SJF) berdasarkan metrik utama seperti waktu tunggu rata-rata, waktu penyelesaian rata-rata, dan *throughput*. RQ2 akan mengeksplorasi kondisi spesifik, seperti jenis beban kerja, lingkungan komputasi (misalnya, kondisi *preemptive* atau non-*preemptive*), serta variasi *quantum time* RR, di mana salah satu algoritma tersebut menunjukkan performa yang lebih unggul. Terakhir, RQ3 bertujuan untuk mengidentifikasi secara konsisten kelebihan dan kekurangan utama dari masing-masing algoritma berdasarkan temuan dari studi-studi komparatif yang dianalisis.

2. METODE

2.1. Desain Kajian

Penelitian ini menggunakan pendekatan *Systematic Literature Review* (SLR) untuk mengidentifikasi, mengevaluasi, dan menyintesis bukti-bukti penelitian yang relevan terkait perbandingan kinerja algoritma penjadwalan CPU *Round Robin* (RR) dan *Shortest Job First* (SJF). Metodologi SLR ini disusun berdasarkan pedoman *Preferred Reporting Items for Systematic Reviews and Meta-Analyses* (PRISMA) 2020.

Secara garis besar, alur penelitian ini mengikuti lima tahapan utama yang terstruktur, mulai dari perumusan masalah hingga sintesis data. Tahapan-tahapan tersebut diilustrasikan pada Gambar 1.



Gambar 1. Tahapan Metodologi Penelitian SLR

2.2. Kriteria Kelayakan

Proses seleksi artikel untuk *Systematic Literature Review* (SLR) ini akan mengikuti serangkaian kriteria kelayakan yang telah ditetapkan secara ketat. Kriteria ini dirancang untuk memastikan bahwa hanya studi-studi yang paling relevan dan berkualitas tinggi yang akan disertakan dalam sintesis data. Proses ini dibagi menjadi dua kategori utama: Kriteria Inklusi dan Kriteria Eksklusi.

Tabel 1. Kriteria Inklusi dan Eksklusi

Inklusi	Eksklusi
Artikel membahas atau membandingkan algoritma penjadwalan CPU *Round Robin* (RR) dan *Shortest Job First* (SJF), termasuk varian atau kombinasi dari keduanya (selama ada perbandingan RR dengan SJF).	Artikel bukan merupakan karya ilmiah primer (misalnya, editorial, opini, ulasan buku, *grey literature* seperti *posting* blog atau presentasi tanpa *peer-review*).
Artikel menggunakan pendekatan evaluasi kinerja melalui simulasi atau eksperimen.	Artikel tidak menyajikan evaluasi kinerja kuantitatif atau kualitatif yang jelas (misalnya, hanya berupa deskripsi teoritis tanpa perbandingan).
Artikel merupakan publikasi ilmiah *peer-reviewed* (misalnya, jurnal, prosiding konferensi).	Artikel tidak membahas secara substansial algoritma RR dan SJF (misalnya, hanya menyebutkan secara sepintas).
Artikel diterbitkan dalam rentang tahun 2020 hingga 2024 (untuk memastikan relevansi dan kebaruan).	Artikel yang merupakan duplikasi dari studi yang sudah diidentifikasi.
Artikel tersedia dalam bahasa Inggris atau Indonesia.	
Artikel dapat diakses secara penuh (*full-text)*	

Studi-studi yang memenuhi kriteria inklusi kemudian dikelompokkan berdasarkan fokus utamanya, terutama studi komparatif yang membandingkan langsung kinerja RR dan SJF, untuk keperluan sintesis data.

2.3. Sumber Informasi dan Strategi Pencarian

Untuk memastikan cakupan literatur yang komprehensif dan relevan, penelitian ini menggunakan pendekatan sistematis dalam mengidentifikasi artikel. Bagian ini menjelaskan sumber-sumber informasi yang digunakan dan strategi pencarian yang diterapkan untuk mengumpulkan data.

Tabel 2. Sumber Informasi dan Strategi Pencarian

Sumber Informasi	IEEE Xplore
	ScienceDirect
	SpringerLink
	Semantic Scholar
Strategi Pencarian	Strategi pencarian dirancang untuk memaksimalkan cakupan artikel yang relevan. Kata kunci utama yang digunakan melibatkan kombinasi istilah terkait algoritma, konteks, dan metode evaluasi. Contoh *string* pencarian umum yang diadaptasi untuk masing-masing *database* adalah sebagai berikut: ("*Round Robin*" *OR* "RR") *AND* ("*Shortest Job First*" OR "SJF") *AND* ("CPU *Scheduling*" *OR* "Penjadwalan CPU")
	Filter batasan tahun publikasi (2020-2024) dan jenis dokumen (artikel jurnal, prosiding konferensi) diterapkan langsung pada antarmuka pencarian *database* jika tersedia. Tidak ada batasan bahasa yang diterapkan pada tahap pencarian awal, namun seleksi bahasa dilakukan pada tahap penyaringan sesuai kriteria inklusi.

2.4. Proses Seleksi Studi

Proses seleksi studi mengikuti tahapan yang diadaptasi dari diagram alir PRISMA dan dijelaskan pada tabel berikut:

Tabel 3. Proses Seleksi Studi

Identifikasi	Hasil pencarian dari semua *database* digabungkan, dan duplikasi antar *database* dihilangkan menggunakan perangkat lunak manajemen referensi yaitu Mendeley Reference Manager
Penyaringan (*Screening)*	Tahap 1 (Judul dan Abstrak): Judul dan abstrak dari artikel yang teridentifikasi disaring secara independen oleh peneliti tunggal. Setiap artikel dinilai kelayakannya berdasarkan kriteria inklusi dan eksklusi.
	Tahap 2 (Teks Lengkap): Artikel yang lolos tahap pertama kemudian diakses secara penuh (*full-text*). Penilaian kelayakan kedua dilakukan oleh peneliti tunggal. berdasarkan pembacaan teks lengkap untuk memastikan pemenuhan semua kriteria inklusi.
Inklusi Akhir	Artikel yang memenuhi semua kriteria inklusi setelah penilaian teks lengkap dimasukkan dalam SLR ini.

2.5. Proses Pengumpulan Data

Untuk setiap studi yang disertakan, proses ekstraksi data dilakukan secara sistematis oleh peneliti menggunakan formulir ekstraksi data standar yang telah diuji coba. Guna meminimalkan risiko bias subjektif, peneliti menerapkan protokol verifikasi dua tahap. Tahap pertama adalah sesi ekstraksi data awal dari semua artikel terpilih. Setelah jeda waktu yang cukup untuk menjaga objektivitas, peneliti melakukan tahap kedua yaitu sesi verifikasi, di mana seluruh data yang telah diekstraksi diperiksa ulang secara cermat dengan merujuk kembali ke artikel sumber. Proses ini bertujuan untuk memastikan tidak ada kesalahan interpretasi atau pencatatan data sebelum dilanjutkan ke tahap sintesis. Item data utama yang dikumpulkan dari setiap studi, yang disajikan dalam Tabel 4.

Tabel 4. Item Ekstraksi Data dari Setiap Studi

Informasi bibliografi (penulis, tahun terbit, judul, sumber publikasi).
Tujuan penelitian dari studi tersebut.
Algoritma penjadwalan spesifik yang dibahas/dibandingkan (misalnya, RR standar, varian RR dengan *quantum* dinamis, SJF *preemptive*, SJF non-*preemptive*, kombinasi RR-SJF).
Parameter kunci algoritma (misalnya, nilai *quantum time* untuk RR, kriteria prioritas jika ada).
Karakteristik beban kerja yang digunakan (misalnya, jumlah proses, pola waktu kedatangan, distribusi waktu layanan/*burst time*, apakah homogen atau heterogen).
Metode evaluasi yang digunakan (misalnya, jenis simulasi, alat simulator, parameter simulasi, atau detail setup eksperimen jika ada).
Metrik kinerja utama yang dievaluasi dan didefinisikan dalam studi tersebut, terutama waktu tunggu rata-rata (*Average Waiting Time*/AWT), Waktu Penyelesaian Rata-rata (*Average Turnaround Time*/ATAT), dan *Throughput*
Hasil utama dan kesimpulan studi terkait perbandingan kinerja RR dan SJF

2.6. Metode Sintesis Data

Mengingat potensi heterogenitas dalam desain studi, parameter, dan pelaporan hasil di antara artikel yang disertakan, meta-analisis statistik formal tidak direncanakan. Sebaliknya, sintesis naratif (deskriptif) akan digunakan untuk merangkum dan membandingkan temuan dari studi-studi yang disertakan.

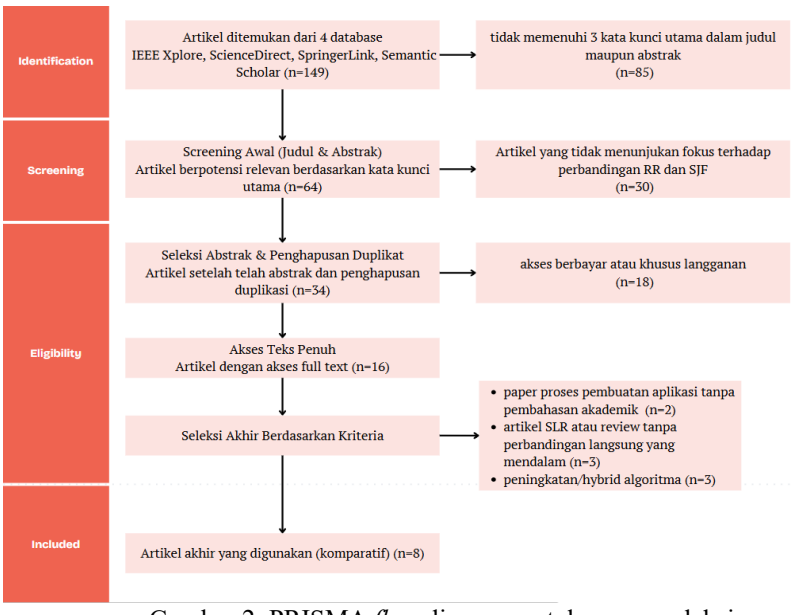
Proses sintesis akan melibatkan:

1. Tabulasi karakteristik utama dan temuan dari setiap studi yang disertakan.
2. Pengelompokan temuan berdasarkan pertanyaan penelitian (RQ1, RQ2, RQ3).
3. Identifikasi pola konsisten, perbedaan, atau kontradiksi dalam temuan antar studi.
4. Analisis kualitatif untuk mengeksplorasi kemungkinan alasan di balik heterogenitas hasil, seperti perbedaan dalam metodologi, jenis beban kerja, atau varian algoritma yang dievaluasi.

Ukuran efek yang akan disajikan terutama adalah nilai-nilai metrik kinerja yang dilaporkan (misalnya, AWT dalam milidetik, ATAT dalam milidetik, *throughput* dalam proses/detik) atau perbandingan relatif (misalnya, persentase perbedaan) antara algoritma RR dan SJF jika dilaporkan demikian oleh studi primer.

3. HASIL

3.1. Seleksi Studi



Gambar 2. PRISMA *flow* diagram untuk proses seleksi

3.2. Hasil Studi Individual

Bagian ini menyajikan temuan kunci dari masing-masing 8 studi yang dianalisis terkait perbandingan kinerja algoritma *Round Robin* (RR) dan *Shortest Job First* (SJF), serta variannya.

Tabel 5. Daftar Studi yang Dianalisis

No.	Artikel	Referensi
1	*Comparative Study: Preemptive Shortest Job First and Round Robin Algorithms*	[8]
2	*Analisis perbandingan algoritma penjadwalan round robin dan shortest job first untuk manajemen proses dalam single processing*	[9]
3	*Meningkatkan responsivitas pada sistem operasi android melalui implementasi algoritma penjadwalan mutakhir*	[10]
4	*Simulation of cpu scheduling algorithms using poisson distribution*	[11]
5	*Performance assessment of CPU scheduling algorithms: A scenario-based approach with FCFS, RR, and SJF*	[12]
6	*Estimation of CPU Scheduling Algorithms Efficiency Using Object Oriented Programming*	[13]
7	*Developing a platform using petri nets and gpensim for simulation of multiprocessor scheduling algorithms*	[14]
8	*Efficient Road Traffic Control Using CPU Scheduling Algorithms*	[15]

- Studi 1
  - Temuan Kunci Terkait RQ1 & RQ2:
    - Dalam ketiga studi kasus perbandingan yang disajikan, algoritma *Preemptive* SJF secara konsisten menghasilkan nilai AWT dan ATAT yang lebih rendah dibandingkan algoritma RR [8].
    - Pada studi kasus 1, AWT RR +52% dan ATAT +30% dari *Preemptive* SJF [8].
    - Pada studi kasus 2, AWT RR +52% dan ATAT +35% dari *Preemptive* SJF [8].
    - Pada studi kasus 3, AWT RR +50% dan ATAT +28% dari *Preemptive* SJF [8].
  - Kelebihan/Kekurangan Teridentifikasi (RQ3):
    - *Preemptive* SJF lebih efisien untuk jenis data yang diuji dalam studi ini [8].
    - RR menunjukkan AWT dan ATAT yang lebih tinggi secara signifikan dalam semua skenario yang diuji [8].
- Studi 2
  - Temuan Kunci Terkait RQ1 & RQ2:
    - Secara umum, algoritma *Preemptive* SJF memiliki AWT dan ATAT paling kecil dibandingkan Non-*Preemptive* SJF dan RR dalam dua dari tiga pengujian [9].
    - Pada pengujian ketiga, nilai AWT dan ATAT untuk *Preemptive* SJF dan Non-*Preemptive* SJF sama, disebabkan oleh nilai *arrival time* dan *burst time* proses yang diberikan [9].
    - Algoritma RR, meskipun dengan beberapa nilai *quantum* menunjukkan AWT dan ATAT yang rendah, nilainya tetap lebih tinggi dibandingkan AWT dan ATAT dari algoritma SJF [9].
  - Kelebihan/Kekurangan Teridentifikasi (RQ3):
    - SJF (khususnya *preemptive*) lebih optimal dibandingkan RR dalam hal AWT dan ATAT pada sebagian besar kasus uji [9].
    - Kinerja RR sangat dipengaruhi oleh nilai *quantum time* [9].
- Studi 3
  - Temuan Kunci Terkait RQ1 & RQ2:
    - Algoritma SJF memiliki waktu tunggu terendah (15 ms), sedangkan FCFS tertinggi (50 ms) [10].
    - CFS (dibandingkan dengan RR dan SJF) disebut sebagai opsi terbaik untuk meningkatkan responsivitas dan efisiensi energi di Android, efisien dalam beban kerja ringan maupun berat. SJF juga efisien dalam beban kerja ringan [10].
    - RR disebut mengonsumsi daya tertinggi karena *context switch* [10].
  - Kelebihan/Kekurangan Teridentifikasi (RQ3):
    - SJF unggul dalam waktu tunggu dan *throughput* pada beban kerja rendah [10].
    - RR menawarkan kinerja seimbang dan responsivitas tinggi jika ukuran *quantum* diatur dengan benar, namun bisa boros daya [10].
    - FCFS sering menghasilkan waktu tunggu lama dan *throughput* rendah pada beban kerja besar [10].
- Studi 4
  - Temuan Kunci Terkait RQ1 & RQ2:
    - SJF Non-*Preemptive* menghasilkan AWT dan ATT minimal dibandingkan DRR dan HRRN untuk *dataset* yang dihasilkan menggunakan Distribusi *Poisson* [11].

        - Untuk data yang diuji: SJF (AWT 313.67ms, ATT 457.17ms), DRR (AWT 388.83ms, ATT 532.83ms), HRRN (AWT 333.33ms, ATT 476.83ms) [11].
    - Kelebihan/Kekurangan Teridentifikasi (RQ3):
        - SJF Non-*Preemptive* menunjukkan kinerja terbaik dalam mengurangi AWT dan ATT dalam simulasi ini [11].
- Studi 5
    - Temuan Kunci Terkait RQ1 & RQ2:
        - SJF secara konsisten memberikan kinerja superior (waktu respons dan *turnaround time* rata-rata lebih rendah) di kelima skenario yang diuji karena memprioritaskan proses pendek [12].
        - Kinerja FCFS bervariasi; baik jika proses pendek datang lebih dulu atau *burst time* seragam, buruk jika proses panjang datang lebih dulu [12].
        - Kinerja RR sangat bergantung pada ukuran *quantum*; *quantum* kecil (1 unit) berkinerja buruk karena *overhead context switching* tinggi. *Quantum* lebih besar (5 atau 10 unit) membuat kinerja RR mendekati FCFS, terutama dengan *burst time* pendek [12].
    - Kelebihan/Kekurangan Teridentifikasi (RQ3):
        - SJF paling efisien jika *burst time* diketahui [12].
        - FCFS sederhana namun rentan terhadap *convoy effect* [12].
        - RR adil namun efisiensinya sangat ditentukan oleh ukuran *quantum*; *quantum* terlalu kecil menyebabkan *overhead*, terlalu besar menjadi seperti FCFS [12].
- Studi 6
    - Temuan Kunci Terkait RQ1 & RQ2:
        - SRTF, diikuti SJF, memberikan kinerja terbaik dalam hal AWT dan ATT pada sebagian besar kasus, baik untuk waktu kedatangan yang sama maupun berbeda [13].
        - RR menunjukkan kinerja terburuk untuk AWT dan ATT [13].
        - Namun, RR terbukti terbaik untuk ART [13].
        - Utilisasi CPU hampir sama (100% atau mendekati) untuk semua algoritma. *Throughput* sedikit lebih baik untuk RR pada beberapa kasus dengan waktu kedatangan berbeda [13].
    - Kelebihan/Kekurangan Teridentifikasi (RQ3):
        - SRTF/SJF unggul untuk AWT dan ATT [13].
        - RR unggul untuk ART tetapi buruk untuk AWT/ATT [13].
- Studi 7
    - Temuan Kunci Terkait RQ1 & RQ2:
        - FCFS dan SJF memiliki *turnaround time* rata-rata lebih rendah karena tidak ada c*ontext switching* antar *task* yang sama. SJF menunjukkan *outlier* signifikan pada *turnaround time* karena penundaan proses panjang [14].
        - RR, MLFQ, dan CFS memiliki *turnaround time* rata-rata yang lebih sebanding [14].
        - MLFQ dan CFS memiliki *response time* rata-rata lebih baik daripada FCFS, SJF, dan RR karena mekanisme prioritas/*vruntime* yang memungkinkan *job* baru berjalan lebih awal [14].
        - SJF dapat menjadi *benchmark* optimal untuk *turnaround time* jika semua *job* datang bersamaan [14].
    - Kelebihan/Kekurangan Teridentifikasi (RQ3):
        - SJF: *Turnaround time* baik tetapi bisa ada kelaparan (*starvation*) untuk *job* panjang (terlihat dari *outlier*) dan butuh pengetahuan *burst time* [14].
        - RR: Adil, namun bisa memiliki *throughput* rendah dan *response time* kurang optimal pada antrean panjang [14].
        - FCFS: Linearitas pada *turnaround time*, buruk untuk *response time* jika ada *job* panjang di depan [14].
- Studi 8
    - Temuan Kunci Terkait RQ1 & RQ2 (dalam konteks lalu lintas):
        - SJFCS dan SJF adalah yang paling efisien dalam mengurangi TWT kendaraan [15].
        - SJFCS memiliki TWT rata-rata terendah (M=2361.04 detik), diikuti SJF (M=2421.32 detik). RR memiliki TWT rata-rata tertinggi (M=4595.36 detik) dan paling tidak efisien [15].
        - Tidak ada perbedaan statistik signifikan antara TWT rata-rata SJFCS dan SJF [15].
    - Kelebihan/Kekurangan Teridentifikasi (RQ3) (dalam konteks lalu lintas):
        - SJF dan SJFCS lebih efisien dan konsisten daripada LQF dan RR [15].
        - Modifikasi pada SJF untuk menjadi SJFCS (mempertimbangkan *context-switch time* sinyal) tidak memberikan peningkatan efisiensi yang signifikan secara statistik [15].
        - RR adalah yang paling tidak efisien [15].

3.3. Hasil Sintesis

Bagian ini menyajikan sintesis naratif dari temuan-temuan kunci yang diekstrak dari delapan studi komparatif yang dianalisis. Sintesis ini bertujuan untuk menjawab pertanyaan penelitian yang telah dirumuskan.

3.3.1. Perbandingan Kinerja Umum Algoritma *Round Robin* (RR) dan *Shortest Job First* (SJF) (RQ1)

Pertanyaan penelitian pertama (RQ1) bertujuan untuk membandingkan kinerja algoritma *Round Robin* (RR) dan *Shortest Job First* (SJF) dalam hal waktu tunggu rata-rata (AWT), waktu penyelesaian rata-rata (ATAT), dan *throughput*.

- Waktu Tunggu Rata-rata (AWT) dan Waktu Penyelesaian Rata-rata (ATAT): Sebagian besar studi yang dianalisis secara konsisten menunjukkan bahwa algoritma *Shortest Job First* (SJF), terutama dalam varian *preemptive* (juga dikenal sebagai *Shortest Remaining Time First* - SRTF), cenderung menghasilkan AWT dan ATAT yang lebih rendah dibandingkan algoritma *Round Robin* (RR).

Tabel 6. Ringkasan Temuan Terkait AWT dan ATAT (RQ1)

Peneliti & Tahun	Temuan Utama
Purnomo & Putra (2024)	AWT pada RR 50-52% lebih tinggi dan ATAT 28-35% lebih tinggi dibandingkan *Preemptive* SJF.
Taufiq et al. (2021)	*Preemptive* SJF memiliki AWT dan ATAT paling kecil. RR, meskipun bisa mencapai nilai rendah, tetap lebih tinggi dari SJF.
Mishra & Ahmed (2020)	Non-*Preemptive* SJF menghasilkan AWT dan ATT yang paling minimal.
Hajjar et al. (2024)	SJF secara konsisten menunjukkan kinerja superior (AWT & ATAT lebih rendah). RR dengan *quantum* kecil berkinerja buruk.
Tufegdžić et al. (2022)	SRTF (varian SJF) dan SJF memberikan performa terbaik untuk AWT dan ATT. RR adalah yang terburuk.
Kampani (2021)	Dalam konteks kontrol lalu lintas, SJF dan variannya lebih efisien dalam mengurangi Total Waktu Tunggu (TWT).

- *Throughput*: Temuan terkait *throughput* lebih bervariasi.

Tabel 7. Ringkasan Temuan Terkait Throughput (RQ1)

Peneliti & Tahun	Temuan Utama (*Throughput*)
Rahman et al. (2024)	SJF unggul dalam *throughput* pada situasi beban kerja rendah.
Tufegdžić et al. (2022)	*Throughput* sedikit lebih baik untuk RR pada beberapa kasus, meskipun utilisasi CPU hampir sama.
Dirdal et al. (2024)	FCFS dan SJF memiliki *turnaround time* lebih rendah (implikasi *throughput* per proses lebih baik untuk *job* cepat) karena tidak ada *context switching* internal. Namun, *throughput* SJF bisa terdistorsi karena *starvation*.

- Waktu Respons (*Response Time*):

Tabel 8. Ringkasan Temuan Terkait Waktu Respons (RQ1)

Peneliti & Tahun	Temuan Utama (Waktu Respons)
Hajjar et al. (2024)	Tidak secara spesifik membandingkan waktu respons, tetapi mencatat kinerja RR yang buruk secara umum dengan *quantum* kecil.
Tufegdžić et al. (2022)	RR secara eksplisit terbukti menjadi yang terbaik untuk Waktu Respons Rata-rata (ART).
Dirdal et al. (2024)	MLFQ dan CFS memiliki ART lebih baik daripada SJF dan RR. Sifat antrean FIFO pada RR dapat menunda respons untuk proses yang datang belakangan.

Tabel 9. Ringkasan Temuan RQ1

Metrik Kinerja	Temuan Utama
Waktu Tunggu (AWT) & Waktu Penyelesaian (ATAT)	SJF/SRTF secara konsisten lebih unggul, menghasilkan nilai AWT dan ATAT yang jauh lebih rendah dibandingkan RR di sebagian besar skenario yang diuji.
Throughput	Hasil bervariasi. SJF cenderung unggul pada beban kerja ringan, sementara RR dapat menunjukkan *throughput* sedikit lebih baik pada kasus dengan waktu kedatangan proses yang berbeda-beda.
Waktu Respons (*Response Time*)	RR menunjukkan potensi lebih baik, terutama pada metrik *Average Response Time* (ART), yang menjadikannya pilihan utama untuk sistem interaktif.

3.3.2. Kondisi Performa Terbaik untuk RR atau SJF (RQ2)

Pertanyaan penelitian kedua (RQ2) bertujuan untuk mengidentifikasi jenis beban kerja dan lingkungan komputasi di mana RR atau SJF menunjukkan performa terbaik.

Tabel 10. Kondisi Performa Superior untuk SJF/SRTF dan RR (RQ2)

Performa Terbaik SJF/SRTF	SJF/SRTF secara konsisten menunjukkan performa terbaik (AWT dan ATAT terendah) ketika durasi eksekusi (*burst time*)proses diketahui atau dapat diestimasi dengan baik [12], [13].
	Keunggulannya lebih menonjol pada kondisi beban kerja dengan variasi *burst time* yang tinggi, di mana SJF/SRTF dapat secara efektif mendahulukan proses-proses pendek [12].
	Rahman et al. (2024) menyebutkan SJF efisien dalam beban kerja ringan di OS Android [10].
	Dirdal et al. (2024) menyatakan SJF bisa menjadi *benchmark* optimal untuk *turnaround time* jika semua pekerjaan datang bersamaan [14].
Performa Terbaik RR	RR dirancang untuk sistem *time-sharing* dan interaktif, di mana waktu respons yang cepat lebih diutamakan daripada meminimalkan AWT/ATAT keseluruhan. Studi oleh Tufegdžić et al. (2022) mendukung ini dengan menunjukkan RR memiliki ART terbaik [12], [13].
	Kinerja RR sangat dipengaruhi oleh pemilihan ukuran *time quantum*. *Quantum* yang terlalu kecil menyebabkan *overhead context switching* yang tinggi dan menurunkan kinerja. *Quantum* yang terlalu besar akan membuat RR berperilaku seperti FCFS, yang dapat memperburuk waktu respons untuk proses-proses pendek yang datang belakangan [12].
	Hajjar et al. (2024) menemukan bahwa ketika *quantum* dinaikkan (misalnya ke 5 atau 10 unit), kinerja RR mulai mendekati FCFS, terutama jika *burst time* proses relatif pendek dan waktu kedatangan tidak banyak variasi [12].
	Rahman et al. (2024) menyatakan RR menawarkan kinerja seimbang dan responsivitas tinggi jika ukuran *quantum* diatur dengan benar [10].

Tabel 11. Ringkasan Temuan RQ2

Algoritma	Kondisi Performa Superior (Sintesis)
Shortest Job First (SJF/SRTF)	Menunjukkan kinerja paling efisien pada beban kerja dengan variasi *burst time* yang tinggi, pada beban kerja ringan, atau ketika durasi eksekusi proses dapat diestimasi secara akurat. Algoritma ini menjadi *benchmark* optimal jika semua pekerjaan tiba secara bersamaan.
Round Robin (RR)	Unggul dalam lingkungan sistem *time-sharing* dan interaktif yang memprioritaskan waktu respons yang cepat dan merata untuk semua proses. Performanya sangat bergantung pada pemilihan ukuran *time quantum* yang tepat untuk menyeimbangkan responsivitas dan *overhead*.

3.3.3. Kelebihan dan Kekurangan RR dan SJF (RQ3)

Pertanyaan penelitian ketiga (RQ3) bertujuan untuk merangkum kelebihan dan kekurangan masing-masing algoritma berdasarkan temuan dari studi-studi yang dianalisis.

Tabel 12. Kelebihan dan Kekurangan Algoritma SJF/SRTF dan RR (RQ3)

Shortest Job First (SJF)	Kelebihan	Secara umum menghasilkan waktu tunggu rata-rata (AWT) dan waktu penyelesaian rata-rata (ATAT) yang paling minimal, terutama varian *preemptive* (SRTF). Ini menjadikannya optimal untuk metrik tersebut jika *burst time* diketahui [9], [11], [12], [13].
		Efisien untuk beban kerja ringan dan ketika proses-proses pendek dapat didahulukan [10].
	Kekurangan	Kesulitan utama adalah kebutuhan untuk mengetahui atau mengestimasi *burst time* proses di masa depan secara akurat, yang sering kali tidak praktis dalam sistem nyata [9].
		Potensi terjadinya *starvation* untuk proses-proses dengan *burst time* yang panjang, karena proses-proses pendek akan selalu didahulukan jika terus berdatangan. Dirdal et al. (2024) mengamati *outlier* signifikan pada *turnaround time* SJF yang disebabkan oleh penundaan eksekusi proses panjang [14].
		Kurang ideal untuk sistem interaktif di mana waktu respons yang cepat untuk semua pengguna lebih penting daripada AWT/ATAT terendah secara keseluruhan [14].
		Menjamin keadilan (*fairness*) karena setiap proses mendapatkan jatah waktu CPU secara bergantian, sehingga mencegah terjadinya *starvation* absolut [8].

Round Robin (RR)	Kelebihan	Cocok untuk sistem *time-sharing* dan lingkungan interaktif karena memberikan waktu respons yang relatif baik untuk semua proses, terutama jika dibandingkan dengan FCFS atau SJF dalam beberapa aspek responsivitas awal [13].
		Sederhana untuk diimplementasikan [8].
	Kekurangan	Kinerja sangat sensitif terhadap pemilihan ukuran *time quantum*. Jika *quantum* terlalu kecil, terjadi *overhead* tinggi akibat seringnya *context switching*. Jika *quantum* terlalu besar, RR berperilaku seperti FCFS dan waktu respons untuk proses-proses pendek bisa menjadi buruk [9], [12].
		AWT dan ATAT umumnya lebih tinggi dibandingkan dengan SJF/SRTF pada sebagian besar skenario yang diuji dalam studi-studi komparatif [8], [9], [13].
		Dapat memiliki *throughput* yang lebih rendah pada beberapa kondisi, dan konsumsi daya lebih tinggi karena *context switching* [10].

Tabel 13. Ringkasan Temuan RQ3

Algoritma	Kelebihan	Kekurangan
*Shortest Job First (*SJF/SRTF*)*	AWT & ATAT terendah (paling optimal)	Waktu respons buruk, tidak cocok untuk sistem interaktif.
	Efisien pada beban kerja ringan.	Risiko *starvation* untuk proses panjang.
		Sulit memprediksi *burst time* secara akurat.
*Round Robin (*RR*)*	Menjamin keadilan (*fairness*).	*Overhead* tinggi akibat seringnya *context switching.*
	Waktu respons cepat, ideal untuk sistem interaktif.	AWT & ATAT lebih tinggi dibanding SJF.
	Memiliki implementasi yang sederhana.	Performa sensitif terhadap ukuran *time quantum*

3.4. Penilaian Bias Pelaporan dan Kepastian Bukti

Bias Pelaporan: Dalam tinjauan ini, tidak dilakukan analisis statistik formal untuk menilai bias pelaporan di antara studi-studi yang disertakan. Namun, semua 8 studi yang dianalisis merupakan publikasi *peer-reviewed* yang diakses dari *database* akademik terkemuka. Hal ini diasumsikan dapat mengurangi risiko bias publikasi secara signifikan, meskipun tetap ada kemungkinan bahwa studi dengan hasil negatif atau non-signifikan kurang terwakili dalam literatur yang tersedia.

4. DISKUSI

Tinjauan sistematis ini memberikan gambaran yang konsisten mengenai *trade-off* fundamental antara algoritma *Round Robin* (RR) dan *Shortest Job First* (SJF). Temuan utama mengonfirmasi bahwa SJF, terutama varian *preemptive*-nya (SRTF), unggul dalam metrik efisiensi seperti AWT dan ATAT, yang menjadikannya superior dalam skenario yang mengutamakan *throughput* dan penyelesaian tugas secepat mungkin. Di sisi lain, RR tetap menjadi pilihan relevan untuk sistem interaktif yang memprioritaskan keadilan (*fairness*) dan waktu respons yang merata, meskipun harus mengorbankan efisiensi secara keseluruhan. Konsistensi temuan ini di berbagai studi [8], [12], [13] menunjukkan bahwa *trade-off* ini bersifat inheren pada desain kedua algoritma.

4.1. Keterbatasan Penelitian

Meskipun tinjauan ini dilakukan secara sistematis, terdapat beberapa keterbatasan yang perlu diperhatikan. Pertama, sintesis dilakukan secara naratif karena tingginya heterogenitas dalam metodologi, parameter simulasi, dan beban kerja yang digunakan oleh studi-studi primer. Hal ini menghalangi dilakukannya meta-analisis statistik formal untuk mendapatkan ukuran efek kuantitatif yang tunggal. Kedua, kajian ini rentan terhadap bias publikasi, di mana studi dengan hasil yang signifikan atau positif lebih mungkin untuk dipublikasikan dibandingkan studi dengan hasil non-signifikan. Ketiga, fokus eksklusif pada perbandingan RR dan SJF membatasi generalisasi temuan pada algoritma penjadwalan yang lebih modern, seperti *Completely Fair Scheduler* (CFS) yang disinggung dalam beberapa studi yang dianalisis[10], [14].

4.2. Implikasi dan Arah Penelitian Selanjutnya

Hasil dari tinjauan ini memiliki beberapa implikasi praktis dan membuka peluang untuk penelitian di masa depan.

Tabel 14. Implikasi Praktis dan Arah Penelitian Selanjutnya

Topik	Detail / Rekomendasi
Implikasi Praktis	Pemilihan algoritma harus didasarkan pada prioritas sistem: - Sistem *Batch* (Efisiensi): Pilih SJF jika *burst time* dapat diestimasi. - Sistem Interaktif (Responsivitas): Pilih RR untuk respons yang adil dan cepat.
Arah Penelitian Selanjutnya	Fokus riset di masa depan dapat mencakup: - Algoritma Hibrida: Gabungkan keadilan RR dengan efisiensi SJF.

	- Evaluasi Beban Kerja Modern: Uji algoritma pada platform *Cloud* dan *IoT*. - Analisis Prediksi *Burst Time*: Kaji dampak ketidakakuratan prediksi *burst time* pada kinerja SJF dibandingkan stabilitas RR.

5. KESIMPULAN

Kajian literatur sistematis ini dilakukan untuk menganalisis dan membandingkan kinerja algoritma penjadwalan CPU *Round Robin* (RR) dan *Shortest Job First* (SJF) berdasarkan delapan studi komparatif yang diterbitkan antara tahun 2020 dan 2024. Tujuan utama adalah untuk mengidentifikasi keunggulan, kelemahan, serta kondisi optimal bagi masing-masing algoritma berdasarkan metrik kinerja seperti waktu tunggu rata-rata (AWT), waktu penyelesaian rata-rata (ATAT), dan *throughput*.

Hasil analisis menunjukkan bahwa algoritma *Shortest Job* First (SJF), terutama dalam varian *preemptive* (SRTF), secara umum lebih unggul dalam menghasilkan AWT dan ATAT yang lebih rendah dibandingkan algoritma *Round Robin* dalam berbagai skenario simulasi yang diuji. Keunggulan SJF ini terutama terlihat pada beban kerja dengan variasi *burst time* yang signifikan dan ketika *burst time* proses dapat diketahui atau diestimasi dengan akurat. Namun, SJF memiliki kelemahan inheren terkait potensi *starvation* untuk proses-proses panjang dan kesulitan praktis dalam memprediksi *burst time* di lingkungan nyata.

Di sisi lain, algoritma *Round Robin* (RR) tetap menjadi pilihan yang relevan untuk sistem interaktif dan *time-sharing* karena kemampuannya memberikan keadilan dan waktu respons yang lebih merata bagi semua proses. Meskipun demikian, kinerja RR sangat dipengaruhi oleh pemilihan ukuran *time quantum*; *quantum* yang terlalu kecil dapat meningkatkan *overhead* akibat *context switching*, sedangkan *quantum* yang terlalu besar dapat membuatnya berperilaku seperti FCFS dan memperburuk waktu respons. Secara umum, AWT dan ATAT pada RR cenderung lebih tinggi dibandingkan SJF.

Tidak ada algoritma tunggal yang optimal untuk semua kondisi. Pemilihan antara RR dan SJF harus didasarkan pada pertimbangan cermat terhadap prioritas metrik kinerja sistem dan karakteristik beban kerja yang dihadapi. Temuan dari tinjauan ini menggarisbawahi pentingnya pemahaman mendalam terhadap kedua algoritma ini bagi para pengembang sistem dan mengidentifikasi kebutuhan penelitian lebih lanjut, terutama dalam eksplorasi algoritma hibrida dan evaluasi pada lingkungan sistem nyata dengan beban kerja modern yang kompleks.

REFERENSI

[1] A. Silberschatz, P. B. Galvin, and G. Gagne, *Operating system concepts essentials*. Wiley Publishing, 2013.

[2] A. S. Tanenbaum and H. Bos, *Modern operating systems*, vol. 2. Pearson Education, Inc., 2014. [Online].

[3] W. Stallings, *Operating systems: Internals and design principles*, vol. 68. Prentice Hall Press, 2008. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/21820730

[4] S. Negi and P. Kalra, "A comparative performance analysis of various variants of round robin scheduling algorithm," in *International Journal of Information & Computation Technology*, 2014, pp. 765–772. [Online]. Available: https://api.semanticscholar.org/CorpusID:17492148

[5] Y. A. Adekunle, Z. O. Ogunwobi, A. S. Jerry, B. T. Efuwape, S. Ebiesuwa, and J.-P. Ainam, "A comparative study of scheduling algorithms for multiprogramming in real-time systems," *Int. J. Innov. Sci. Res.*, vol. 12, no. 1, pp. 180–185, 2014, [Online]. Available: http://www.ijisr.issr-journals.org/

[6] S. Kitchenham, B., & Charters, "Guidelines for performing systematic literature reviews in software engineering," *Tech. report, Ver. 2.3 EBSE Tech. Report. EBSE*, no. January 2007, pp. 1–57, 2007.

[7] P. Singh, V. Singh, and A. Pandey, "Analysis and comparison of CPU scheduling algorithms," *Int. J. Emerg. Technol. Adv. Eng.*, vol. 4, no. 1, pp. 91–95, 2014.

[8] R. Purnomo and T. D. Putra, "Comparative study: Preemptive shortest job first and round robin algorithms," *Sink. J. dan Penelit. Tek. Inform.*, vol. 8, no. 2, pp. 756–763, 2024.

[9] L. O. M. Taufiq, L. F. Aksara, and M. Yamin, "Analisis perbandingan algoritma penjadwalan round robin dan shortest job first untuk manajemen proses dalam single processing," *semanTIK*, 2021, [Online].

[10] R. Rahman, S. N. D. Fortuna, R. Triansyah, and M. A. Fahrezi, "Meningkatkan responsivitas pada sistem operasi android melalui implementasi algoritma penjadwalan mutakhir," *Router J. Tek. Inform. dan Terap.*, vol. 2, no. 3, pp. 56–65, 2024.

[11] A. Mishra and A. O. Ahmed, "Simulation of CPU scheduling algorithms using poisson distribution," *Int. J. Wirel. Microw. Technol.*, vol. 6, no. 2, pp. 71–78, 2020.

[12] O. Hajjar, E. Mekhallalati, N. Annwty, F. Alghayadh, I. Keshta, and M. Algabri, "Performance assessment of CPU scheduling algorithms: A scenario-based approach with FCFS, RR, and SJF," *J. Comput. Sci.*, vol. 20, no. 9, pp. 972–985, 2024.

[13] M. Tufegdžić, V. Jevremović, and Z. Petrović, "Estimation of CPU scheduling algorithms efficiency using object oriented programming," *quantum*, vol. 1, p. 4, 2022.

[14] D. O. Dirdal, D. Vo, Y. Feng, and R. Davidrajuh, "Developing a platform using petri nets and gpensim for simulation of multiprocessor scheduling algorithms," *Appl. Sci.*, vol. 14, no. 13, p. 5690, 2024.

[15] S. Kampani, "Efficient road traffic control using CPU scheduling algorithms," *Int. J. Sci. Res. Publ.*, vol. 11, no. 6, pp. 628–651, 2021, doi: 10.29322/ijsrp.11.06.2021.p11483.